Collisions and Semi-Free-Start Collisions for Round-Reduced RIPEMD-160

Fukang Liu[1], Florian Mendel[2], and Gaoli Wang[1](✉)

[1] Shanghai Key Laboratory of Trustworthy Computing, School of Computer Science and Software Engineering, East China Normal University, Shanghai, China
liufukangs@163.com, glwang@sei.ecnu.edu.cn
[2] Graz University of Technology, Graz, Austria
florian.mendel@iaik.tugraz.at

Abstract. In this paper, we propose an improved cryptanalysis of the double-branch hash function RIPEMD-160 standardized by ISO/IEC. Firstly, we show how to theoretically calculate the step differential probability of RIPEMD-160, which was stated as an open problem by Mendel *et al.* at ASIACRYPT 2013. Secondly, based on the method proposed by Mendel *et al.* to automatically find a differential path of RIPEMD-160, we construct a 30-step differential path where the left branch is sparse and the right branch is controlled as sparse as possible. To ensure the message modification techniques can be applied to RIPEMD-160, some extra bit conditions should be pre-deduced and well controlled. These extra bit conditions are used to ensure that the modular difference can be correctly propagated. This way, we can find a collision of 30-step RIPEMD-160 with complexity 2^{67}. This is the first collision attack on round-reduced RIPEMD-160. Moreover, by a different choice of the message words to merge two branches and adding some conditions to the starting point, the semi-free-start collision attack on the first 36-step RIPEMD-160 from ASIACRYPT 2013 can be improved. However, the previous way to pre-compute the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ costs too much. To overcome this obstacle, we are inspired by Daum's *et al.* work on MD5 and describe a method to reduce the time complexity and memory complexity to pre-compute that equation. Combining all these techniques, the time complexity of the semi-free-start collision attack on the first 36-step RIPEMD-160 can be reduced by a factor of $2^{15.3}$ to $2^{55.1}$.

Keywords: RIPEMD-160 · Semi-free-start collision · Collision · Hash function · Compression function

1 Introduction

A cryptographic hash function is a function which takes arbitrary long messages as input and output a fixed-length hash value of size n bits. There are three basic requirements for a hash function, which are preimage resistance, second-preimage resistance and collision resistance. Most standardized hash functions are based on the Merkle-Damgård paradigm [2,12] and iterate a compression function H with fixed-size input to compress arbitrarily long messages. Therefore, the compression function itself should satisfy equivalent security requirements so that the hash function can inherit from it. There are two attack models on the compression function. One is called free-start collision attack, the other is semi-free-start collision attack. The free-start collision attack is to find two different pairs of message and chaining value (CV, M), (CV', M') which satisfy $\mathrm{H}(CV, M) = \mathrm{H}(CV', M')$. The semi-free-start collision attack works in the same way apart from an additional condition that $CV = CV'$. The last decade has witnessed the fall of a series of hash functions such as MD4, MD5, SHA-0 and SHA-1 since many break-through results on hash functions cryptanalysis [15,20–23] were obtained. All of these hash functions belong to the MD-SHA family, whose design strategy is based on the utilization of additions, rotations, xor and boolean functions in an unbalanced Feistel network.

RIPEMD family can be considered as a subfamily of the MD-SHA-family since RIPEMD-0 [1] is the first representative and consists of two MD4-like functions computed in parallel with totally 48 steps. The security of RIPEMD-0 was first put into question by Dobbertin [4] and a practical collision attack on it was proposed by Wang *et al.* [20]. In order to reinforce the security of RIPEMD-0, Dobbertin, Bosselaers and Preneel [3] proposed two strengthened versions of RIPEMD-0 in 1996, which are RIPEMD-128 and RIPEMD-160 with 128/160 bits output and 64/80 steps, respectively. In order to make both computation branches more distinct from each other, not only different constants, but also different rotation values, message insertion schedules and boolean functions are used for RIPEMD-128 and RIPEMD-160 in their both branches.

For RIPEMD-128, there has been a series of analysis on it [5,8,16–18], threatening its security. As for RIPEMD-160, Mendel *et al.* [11] proposed an improved method to automatically find the differential path of RIPEMD-160 at ASIACRYPT 2013. With their method, they found a 48-step differential path and a 36-step differential path. Based on the two differential paths, Mendel *et al.* [11] mounted a semi-free-start collision attack on 42-step RIPEMD-160 and a semi-free-start collision attack on the first 36-step RIPEMD-160. Additionally, they also proposed an open problem to theoretically calculate the step differential probability. Besides, there are also some other analytical results on RIPEMD-160, such as a preimage attack [13] on 31-step RIPEMD-160, a distinguisher on up to 51 steps of the compression function [14], a practical semi-free-start collision attack on 36 steps of the compression function [9] (not starting from the first step), and a semi-free-start collision attack on 48-step RIPEMD-160 [19]. However, RIPEMD-160 is yet unbroken and is widely used in the implementations of security protocols as a ISO/IEC standard.

In 2005, Daum investigated the probability computation of T-functions (a function for which the i-th output bit depends only on the i first lower bits of all input words) in his PhD thesis [6]. More specifically, he proposed a method to calculate the probability that T satisfies the equation $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$

where C_0 and C_1 are constants. According to our analysis of the open problem to calculate the step differential probability of RIPEMD-160, we find that calculating such a probability is equivalent to calculating the probability that the modular difference of the internal states is correctly propagated and the bit conditions on the internal states hold. Although Daum's work can be used to calculate the probability that the modular difference is correctly propagated, it can't solve the open problem completely since the probability that one bit condition on the internal state holds is not 1/2 any more. However, by considering the calculation of the probability that T satisfies the equation $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$ from a different perspective, we can deduce some useful characteristics of T which can be used to calculate the probability that the bit conditions hold. In this way, we can solve the open problem completely.

This paper is organized as follows. In Sect. 2, we briefly describe the algorithm of RIPEMD-160. In Sect. 3, we describe our method to calculate the step differential probability. In Sect. 4, we describe our improved way to pre-compute the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$. In Sect. 5, we describe the collision attack on the first 30-step RIPEMD-160. In Sect. 6, we describe the improved semi-free-start collision attack on the first 36-step RIPEMD-160. Finally, we conclude the paper in Sect. 7.

Our Contributions

1. Our method to theoretically calculate the step differential probability consists of two steps. At first, we consider the probability that the modular difference of the internal states holds, which will help obtain some characteristics of Q_i (Q_i is referred to Sect. 2.2). Then, for each characteristics of Q_i, the probability that the bit conditions on the internal states hold under the condition that this characteristic of Q_i holds can be calculated. In this way, the theoretical calculation of the step differential probability of RIPEMD-160 becomes feasible.
2. We deduce a useful property from the PhD thesis of Daum [6]. Based on it, we can convert solving the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ into solving the equation $T^{\lll S_0} \boxplus C_2 = T^{\lll S_1}$. By analyzing the expectation of the number of the solutions to the equation if given many pairs of (C_0, C_1), we can claim that our new method to obtain the solutions at the phase of merging only costs 4 times of checking the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ on average, thus having a negligible influence on the efficiency compared with the previous method [5,11]. Moreover, both the time complexity and memory complexity of our new method to pre-compute the equation is 2^{32}, which is much smaller than the strategy by constructing a table of size 2^{64} to store the solutions.
3. By using the technique described in [11] to automatically find a differential path for RIPEMD-160, we can construct a 30-step differential path where the left branch is sparse and the right branch is controlled as sparse as possible. For the left branch, we leave it holding probabilistically. For the right branch, we apply the message modification techniques [20] to it. However, according to our analysis of the open problem to theoretically calculate the step

differential probability of RIPEMD-160, the differential path of RIPEMD-160 holds only when both the bit conditions and the modular difference of the internal states hold. That's different from MD4 since the differential path of MD4 holds only when the bit conditions on the internal states hold. Since the message modification can only be used to ensure the bit conditions hold, the difficulty is how to have the modular difference of the internal states hold when applying it to RIPEMD-160. Fortunately, we discover that we can add some extra bit conditions on the internal states to have the modular difference hold. Therefore, before applying the message modification, we have to pre-deduce these extra bit conditions on the internal states by considering the characteristics of Q_i. After obtaining the newly added extra bit conditions, by adjusting the message modification techniques so that it can be applied to RIPEMD-160, we can mount a 30-step collision attack on RIPEMD-160 with probability 2^{-67}.

4. Based on the 36-step differential path, by a different choice of message words to merge both branches, we can improve the time complexity of the merging phase. Moreover, based on the characteristics of Q_{15}, we can add some extra bit conditions on Y_{11} at the phase of finding a starting point to further improve our attack. The improved semi-free-start collision attack on the first 36-step RIPEMD-160 is $2^{55.1}$, which is much smaller than the previous best known result (Table 1).

Table 1. Summary of preimage and collision attack on RIPEMD-160.

Target	Attack type	Steps	Complexity	Ref
Comp. function	Preimage	31	2^{148}	[13]
Hash function	Preimage	31	2^{155}	[13]
Comp. function	Semi-free-start collision	36[a]	low	[9]
Comp. function	Semi-free-start collision	36	$2^{70.4}$	[11]
Comp. function	Semi-free-start collision	36	$2^{55.1}$	New
Comp. function	Semi-free-start collision	42[a]	$2^{75.5}$	[11]
Comp. function	Semi-free-start collision	48[a]	$2^{76.4}$	[19]
Hash function	Collision	30	2^{67}	New

[a]An attack starts at an intermediate step.

2 Description of RIPEMD-160

RIPEMD-160 is a 160-bit hash function that uses the Merkle-Damgård construction as domain extension algorithm: the hash function is built by iterating a 160-bit compression function H which takes as input a 512-bit message block M_i and a 160-bit chaining variables CV_i:

$$CV_{i+1} = H(CV_i, M_i)$$

where a message M to hash is padded beforehand to a multiple of 512 bits and the first chaining variable is set to the predetermined initial value IV, that is $CV_0 = IV$. We refer to [3] for a detailed description of RIPEMD-160.

2.1 Notations

For a better understanding of this paper, we introduce the following notations.

1. $\lll$, $\ggg$, $\oplus$, $\vee$, $\wedge$ and $\neg$ represent respectively the logic operation: *rotate left*, *rotate right*, *exclusive or*, *or*, *and*, *negate*.
2. $\boxplus$ and $\boxminus$ represent respectively the modular addition and modular substraction on 32 bits.
3. $M = (m_0, m_1, \ldots, m_{15})$ and $M' = (m'_0, m'_1, \ldots, m'_{15})$ represent two 512-bit message blocks.
4. $\Delta m_i = m'_i - m_i$ represents the modular difference between two message words m_i and m'_i.
5. K^l_j and K^r_j represent the constant used at the left and right branch for round j.
6. Φ^l_j and Φ^r_j represent respectively the 32-bit boolean function at the left and right branch for round j.
7. X_i, Y_i represent respectively the 32-bit internal state of the left and right branch updated during step i for compressing M.
8. X'_i, Y'_i represent respectively the 32-bit internal state of the left and right branch updated during step i for compressing M'.
9. $X_{i,j}$, $Y_{i,j}$ represent respectively the j-th bit of X_i and Y_i, where the least significant bit is the 0th bit and the most significant bit is the 31st bit.
10. Q_i represents the 32-bit temporary state of the right branch updated during step i for compressing M.
11. s^l_i and s^r_i represent respectively the rotation constant used at the left and right branch during step i.
12. $\pi_1(i)$ and $\pi_2(i)$ represent the index of the message word used at the left and right branch during step i.
13. $[Z]_i$ represents the i-th bit of the 32-bit Z.
14. $[Z]_{j\sim i}$ $(0 \le i < j \le 31)$ represents the i-th bit to the j-th bit of the 32-bit word Z.
15. $x_i[j]$, $x_i[-j]$ (x can be X and Y) is the resulting value by only changing the j-th bit of x_i. $x_i[j]$ is obtained by changing the j-th bit of x_i from 0 to 1. $x_i[-j]$ is obtained by changing the j-th bit of x_i from 1 to 0.
16. $x_i[\pm j_1, \pm j_2, \ldots, \pm j_l]$ (x can be X and Y) is the value by changing the j_1-th, j_2-th, j_l-th bit of x_i. The "+" sign means the bit is changed from 0 to 1, and the "−" sign means the bit is changed from 1 to 0.
17. P(A) is the probability of the event A.

2.2 RIPEMD-160 Compression Function

The RIPEMD-160 compression function is a wider version of RIPEMD-128, which is based on MD4, but with the particularity that it consists of two different and almost independent parallel instances of it. We differentiate the two computation branches by left and right branch. The compression function consists of 80 steps divided into 5 rounds of 16 steps each in both branches.

Table 2. Boolean functions and round constants in RIPEMD-160

Round j	ϕ_j^l	ϕ_j^r	K_j^l	K_j^r	Function	Expression
0	XOR	ONX	0x00000000	0x50a28be6	XOR(x, y, z)	$x \oplus y \oplus z$
1	IFX	IFZ	0x5a827999	0x5c4dd124	IFX(x, y, z)	$(x \wedge y) \oplus (\neg x \wedge z)$
2	ONZ	ONZ	0x6ed9eba1	0x6d703ef3	IFZ(x, y, z)	$(x \wedge z) \oplus (y \wedge \neg z)$
3	IFZ	IFX	0x8f1bbcdc	0x7a6d76e9	ONX(x, y, z)	$x \oplus (y \vee \neg z)$
4	ONX	XOR	0xa953fd4e	0x00000000	ONZ(x, y, z)	$(x \vee \neg y) \oplus z$

Initialization. The 160-bit input chaining variable CV_i is divided into five 32-bit words h_i (i = 0, 1, 2, 3, 4), initializing the left and right branch 160-bit internal state in the following way:

$$X_{-4} = h_0^{\ggg 10}, \quad X_{-3} = h_4^{\ggg 10}, \quad X_{-2} = h_3^{\ggg 10}, \quad X_{-1} = h_2, \quad X_0 = h_1.$$
$$Y_{-4} = h_0^{\ggg 10}, \quad Y_{-3} = h_4^{\ggg 10}, \quad Y_{-2} = h_3^{\ggg 10}, \quad Y_{-1} = h_2, \quad Y_0 = h_1.$$

Particularly, CV_0 corresponds to the following five 32-bit words:

$$X_{-4} = Y_{-4} = \texttt{0xc059d148}, X_{-3} = Y_{-3} = \texttt{0x7c30f4b8}, X_{-2} = Y_{-2} = \texttt{0x1d840c95},$$
$$X_{-1} = Y_{-1} = \texttt{0x98badcfe}, X_0 = Y_0 = \texttt{0xefcdab89}.$$

The Message Expansion. The 512-bit input message block is divided into 16 message words m_i of size 32 bits. Each message word m_i will be used once in every round in a permuted order π for both branches.

The Step Function. At round j, the internal state is updated in the following way.

$$\begin{aligned} X_i &= X_{i-4}^{\lll 10} \boxplus (X_{i-5}^{\lll 10} \boxplus \Phi_j^l(X_{i-1}, X_{i-2}, X_{i-3}^{\lll 10}) \boxplus m_{\pi_1(i)} \boxplus K_j^l)^{\lll s_i^l}, \\ Y_i &= Y_{i-4}^{\lll 10} \boxplus (Y_{i-5}^{\lll 10} \boxplus \Phi_j^r(Y_{i-1}, Y_{i-2}, Y_{i-3}^{\lll 10}) \boxplus m_{\pi_2(i)} \boxplus K_j^r)^{\lll s_i^r}, \\ Q_i &= Y_{i-5}^{\lll 10} \boxplus \Phi_j^r(Y_{i-1}, Y_{i-2}, Y_{i-3}^{\lll 10}) \boxplus m_{\pi_2(i)} \boxplus K_j^r, \end{aligned}$$

where i = (1, 2, 3, ..., 80) and j = (0, 1, 2, 3, 4). The details of the boolean functions and round constants for RIPEMD-160 are displayed in Table 2. As for other parameters, you can refer to [3].

The Finalization. A finalization and a feed-forward is applied when all 80 steps have been computed in both branches. The five 32-bit words $h_i^{'}$ composing the output chaining variable are computed in the following way.

$$\begin{aligned} h_0^{'} &= h_1 \boxplus X_{79} \boxplus Y_{78})^{\lll 10}, \\ h_1^{'} &= h_2 \boxplus X_{78}^{\lll 10} \boxplus Y_{77}^{\lll 10}, \\ h_2^{'} &= h_3 \boxplus X_{77}^{\lll 10} \boxplus Y_{76}^{\lll 10}, \\ h_3^{'} &= h_4 \boxplus X_{76}^{\lll 10} \boxplus Y_{80}, \\ h_4^{'} &= h_0 \boxplus X_{80} \boxplus Y_{79}. \end{aligned}$$

3 Calculate the Step Differential Probability

In [11], Mendel *et al.* pointed out that it is not as easy to calculate the differential probability for each step of a given differential path of RIPEMD-160 as that of RIPEMD-128. The main reason is that the step function in RIPEMD-160 is no longer a T-function. Therefore, the accurate calculation of the differential probability becomes very hard. However, we can divide the calculation of the step differential probability into two steps. Define as μ the event that all bit conditions on the internal state hold, as ν the event that the modular difference of the internal state holds. Although Daum has proposed a method [6] to calculate $P(\nu)$, we will use a different method to calculate it, since our goal is not only to calculate $P(\nu)$ but also to obtain some useful characteristics of Q_i. Then, we can leverage the deduced characteristics and the bit conditions on the internal states to calculate $P(\mu\nu)$. In this way, the step differential probability $P(\mu\nu)$ can be obtained. We use the step function of the right branch as an example and give its description below. We will show how to deduce the useful characteristics of Q_i and calculate $P(\mu\nu)$.

3.1 Description of the Open Problem

Since the step function of RIPEMD-160 at both branches has the same form, we take the right branch as an example to describe the open problem.

$$Y_i = Y_{i-4}^{\lll 10} \boxplus (Y_{i-5}^{\lll 10} \boxplus \Phi_j^l(Y_{i-1}, Y_{i-2}, Y_{i-3}^{\lll 10}) \boxplus m_{\pi_2(i)} \boxplus K_j^r)^{\lll s_i^r}.$$

To ensure the given differential path holds, we need to impose conditions on some bits of Y_i and control the modular difference of Y_i. The open problem is how to calculate the probability that both the bit conditions on Y_i and the modular difference of Y_i are satisfied under the condition that all conditions on $Y_{i-1}, Y_{i-2}, Y_{i-3}, Y_{i-4}, Y_{i-5}$ are satisfied. For example, according to the differential path displayed in Table 16, we know that:

$$Y'_{15} = Y_{15}[-5, -20, -26],\ Y'_{14} = Y_{14}[5, 11, 22],\ Y'_{13} = Y_{13}[-9, -24, 26, -30],$$
$$Y'_{12} = Y_{12}[0, -15, 21],\ Y'_{11} = Y_{11}[1, 10, 12, 15, -16, 24, 26, -28],$$
$$Y'_{10} = Y_{10}[-3, 21, 22, 23, 24, 25, 26, -28],\ \Delta m_3 = 0.$$

Firstly, we use Y_{10}, Y_{11}, Y_{12}, Y_{13}, Y_{14}, m_3 to calculate Y_{15}. Then, we use Y'_{10}, Y'_{11}, Y'_{12}, Y'_{13}, Y'_{14}, m'_3 to calculate Y'_{15}. Then, the differential probability for step 15 is equal to the probability that $Y'_{15} = Y_{15} \boxminus 2^5 \boxminus 2^{20} \boxminus 2^{26}$ and that all bit conditions on Y_{15} are satisfied.

3.2 The Probability of $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$

Given two constants C_0 and C_1, Daum has described a method [6] to calculate the probability that T satisfies $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$ $(1 \le S \le 31)$. However, we consider the problem from a different perspective by considering the characteristics of T which satisfies such an equation. In this way, we can not only calculate the probability of this equation, but also can obtain the characteristics of T for further use to theoretically calculate the step differential probability.

Let $R_0||R_1 = T \boxplus C_0$, where R_0 is an S-bit variable representing the higher S bits of $T \boxplus C_0$ and R_1 is a (32-S)-bit variable representing the lower (32-S) bits of $T \boxplus C_0$. Let $R_1'||R_0' = T^{\lll S} \boxplus C_1$, where R_1' is a (32-S)-bit variable representing the higher (32-S) bits of $T^{\lll S} \boxplus C_1$ and R_0' is an S-bit variable representing the lower S bits of $T^{\lll S} \boxplus C_1$. Then, the probability of $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$ $(1 \leq S \leq 31)$ is equal to $\mathrm{P}(R_0 = R_0'$ and $R_1 = R_1')$. Since

$$\begin{aligned}
R_0 &\equiv [T]_{31\sim(32-S)} + [C_0]_{31\sim(32-S)} + carry_0 \; mod \; (2^S),\\
R_0' &\equiv [T]_{31\sim(32-S)} + [C_1]_{(S-1)\sim 0} \; mod \; (2^S),\\
R_1 &\equiv [T]_{(31-S)\sim 0} + [C_0]_{(31-S)\sim 0} \; mod \; (2^{32-S}),\\
R_1' &\equiv [T]_{(31-S)\sim 0} + [C_1]_{31\sim S} + carry_1 \; mod \; (2^{32-S}),
\end{aligned}$$

where $carry_0$ represents the carry from the (31-S)-th bit to the (32-S)-th when calculating $T \boxplus C_0$, and $carry_1$ represents the carry from the (S-1)-th bit to the S-th bit when calculating $T^{\lll S} \boxplus C_1$. For simplicity, we define as κ the event that $carry_0 = 0$ and as ω the event that $carry_1 = 0$. Therefore,

$$\begin{aligned}
&P(R_0 = R_0') = P(\kappa \; and \; [C_0]_{31\sim(32-S)} =\\
&[C_1]_{(S-1)\sim 0}) + P(\overline{\kappa} \; and \; [C_0]_{31\sim(32-S)} + 1 \equiv [C_1]_{(S-1)\sim 0} \; mod \; (2^S)),\\
&P(R_1 = R_1') = P(\omega \; and \; [C_0]_{(31-S)\sim 0} =\\
&[C_1]_{31\sim S}) + P(\overline{\omega} \; and \; [C_0]_{(31-S)\sim 0} \equiv [C_1]_{31\sim S} + 1 \; mod \; (2^{32-S})).
\end{aligned}$$

We denote the positions of the bits of $[C_0]_{(31-S)\sim 0}$ equal to 1 by $k_0, \; k_1, \ldots, k_n$ and denote the positions of the bits of $[C_1]_{(S-1)\sim 0}$ equal to 1 by $r_0, \; r_1, \ldots, r_m$. Then, the value of $P(\kappa)$ and $P(\omega)$ can be directly deduced as below:

1. If $[C_0]_{(31-S)\sim 0} = 0$, then $P(\kappa) = 1$. Otherwise, $P(\kappa) = 1 - \sum_{i=0}^{n} 2^{-(32-S-k_i)}$.
2. If $[C_1]_{(S-1)\sim 0} = 0$, then $P(\omega) = 1$. Otherwise, $P(\omega) = 1 - \sum_{i=0}^{m} 2^{-(S-r_i)}$.

Thus, we can compute $P(R_0 = R_0' \; and \; R_1 = R_1')$ in this way:

1. If $[C_0]_{(31-S)\sim 0} = [C_1]_{31\sim S}$ and $[C_0]_{31\sim(32-S)} = [C_1]_{(S-1)\sim 0}$, then $P(R_0 = R_0' \; and \; R_1 = R_1') = P(\kappa) \times P(\omega)$.
2. If $[C_0]_{(31-S)\sim 0} = [C_1]_{31\sim S}$ and $[C_0]_{31\sim(32-S)} + 1 \equiv [C_1]_{(S-1)\sim 0} \; mod \; (2^S)$, then $P(R_0 = R_0' \; and \; R_1 = R_1') = P(\overline{\kappa}) \times P(\omega)$.
3. If $[C_0]_{(31-S)\sim 0} \equiv [C_1]_{31\sim S} + 1 \; mod \; (2^{32-S})$ and $[C_0]_{31\sim(32-S)} = [C_1]_{(S-1)\sim 0}$, then $P(R_0 = R_0' \; and \; R_1 = R_1') = P(\kappa) \times P(\overline{\omega})$.
4. If $[C_0]_{(31-S)\sim 0} \equiv [C_1]_{31\sim S} + 1 \; mod \; (2^{32-S}))$ and $[C_0]_{31\sim(32-S)} + 1 \equiv [C_1]_{(S-1)\sim 0} \; mod \; (2^S))$, then $P(R_0 = R_0' \; and \; R_1 = R_1') = P(\overline{\kappa}) \times P(\overline{\omega})$.
5. If C_0 and C_1 doesn't belong to any of the above four cases, then $P(R_0 = R_0' \; and \; R_1 = R_1') = 0$.

According to the above method to calculate $P(R_0 = R'_0 \ and \ R_1 = R'_1)$, the following property can be directly deduced. (In fact, we can also deduce it by using the Corollary 4.14 in [6].)

Property 1. Given random constants C_0 and C_1 of 32 bits each, there exists a T of 32 bits which satisfies $(T \boxplus C_0)^{\lll S} = T^{\lll S} \boxplus C_1$ if and only if (C_0, C_1) satisfies one of the following equations:

1. $C_1 = (C_0 \boxminus 1)^{\lll S}$, and $[C_1]_{(S-1)\sim 0} \neq 0$.
2. $C_1 = (C_0 \boxplus 2^{32-S})^{\lll S}$, and $[C_0]_{(31-S)\sim 0} \neq 0$.
3. $C_1 = (C_0 \boxplus 2^{32-S} \boxminus 1)^{\lll S}$, and $[C_1]_{(S-1)\sim 0} \neq 0, [C_0]_{(31-S)\sim 0} \neq 0$.
4. $C_1 = C_0^{\lll S}$.

Example. In the following, we give an example how to calculate the probability of $(T\boxplus$ `0x80bfd9ff`$)^{\lll 12} = T^{\lll 12}\boxplus$ `0xfd9ff80c`. To have a better understanding of our method to calculate the probability, we explain it by Table 3.

Table 3. Calculation of the probability

T	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T																																
C_0	1	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1
	R_0												R_1																			
$T^{\lll 12}$	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20
$T^{\lll 12}$																																
C_1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0
	R'_1																				R'_0											

According to Table 3, we can find the following relationship between C_0 and C_1:

$$[C_0]_{19\sim 0} = [C_1]_{31\sim 12}, \ [C_0]_{31\sim 20} + 1 \equiv [C_1]_{11\sim 0} \ mod \ (2^{12}).$$

Therefore, we can get $P((T\boxplus$ `0x80bfd9ff`$)^{\lll 12} = T^{\lll 12}\boxplus$ `0xfd9ff80c`$) = P(\overline{\kappa}) \times P(\omega)$. By considering the characteristics of T, $P(\overline{\kappa})$ and $P(\omega)$ can be calculated as below:

$$\begin{aligned}
P(\overline{\kappa}) &= P([T]_{19} = 1) + P([T]_{19\sim 18} = 01) + P([T]_{19\sim 17} = 001) \\
&+P([T]_{19\sim 16} = 0001) + P([T]_{19\sim 15} = 00001) + P([T]_{19\sim 14} = 000001) \\
&+P([T]_{19\sim 12} = 00000011) + P([T]_{19\sim 11} = 000000101) \\
&+P([T]_{19\sim 8} = 000000100111) + P([T]_{19\sim 7} = 0000001001101) \\
&+P([T]_{19\sim 6} = 00000010011001) + P([T]_{19\sim 5} = 000000100110001) \\
&+P([T]_{19\sim 4} = 0000001001100001) + P([T]_{19\sim 3} = 00000010011000001) \\
&+P([T]_{19\sim 2} = 000000100110000001) + P([T]_{19\sim 1} = 0000001001100000001) \\
&+P([T]_{19\sim 0} = 00000010011000000001) \\
&= \Sigma_{i=1}^{6} 2^{-i} + 2^{-8} + 2^{-9} + \Sigma_{i=12}^{20} 2^{-i}.
\end{aligned}$$

$$\begin{aligned}
P(\omega) &= 1 - P([T]_{31} = 1) - P([T]_{31-23} = 011111111) - P([T]_{31-22} = 0111111101) \\
&= 1 - (2^{-1} + 2^{-9} + 2^{-10}).
\end{aligned}$$

Thus, $P((T \boxplus \texttt{0x80bfd9ff})^{\lll 12} = T^{\lll 12} \boxplus \texttt{0xfd9ff80c}) \approx 2^{-1}$. In this example, we call $[T]_{19} = 1$ one possible characteristic of T, and we call $[T]_{31} = 1$ one impossible characteristic of T. Totally, there are 17 possible characteristics of T and 3 impossible characteristics of T.

3.3 Calculating the Step Differential Probability

We use the step function of the right branch to explain our method to calculate the step differential probability. Let $\Delta = Y'_i \boxminus Y_i$, $\Delta_{i-5} = Y'^{\lll 10}_{i-5} \boxminus Y^{\lll 10}_{i-5}$, $\Delta_{i-4} = Y'^{\lll 10}_{i-4} \boxminus Y^{\lll 10}_{i-4}$, $\Delta F = \Phi^l_j(Y'_{i-1}, Y'_{i-2}, Y'^{\lll 10}_{i-3}) \boxminus \Phi^l_j(Y_{i-1}, Y_{i-2}, Y^{\lll 10}_{i-3})$, then $P(\nu) = P(\Delta = \Delta_{i-4} \boxplus (\Delta_{i-5} \boxplus \Delta F \boxplus \Delta m_{\pi_2(i)} \boxplus Q_i)^{\lll s^r_i} \boxminus Q_i^{\lll s^r_i})$. Given the differential path and the bit conditions to control the differential propagation, Δ, Δ_{i-5}, Δ_{i-4}, ΔF and $\Delta m_{\pi_2(i)}$ are all fixed. Let $C_0 = \Delta_{i-5} \boxplus \Delta F \boxplus \Delta m_{\pi_2(i)}$ and $C_1 = \Delta \boxminus \Delta_{i-4}$, we can obtain that $P(\nu) = P((Q_i \boxplus C_0)^{\lll s^r_i} = Q_i^{\lll s^r_i} \boxplus C_1)$, which can be quickly calculated as described in Sect. 3.2.

Observe that when calculating Y_i, there are conditions on some bits of Y_{i-4} and Y_i, i.e., some bits of Y_{i-4} and Y_i are fixed. In addition, in order to make the modular difference of Y_i satisfied, there are some constraints on Q_i. By analyzing the constraints carefully, the characteristics of Q_i can be discovered, which will make the theoretical calculation of $P(\mu\nu)$ feasible. By the following example, we will introduce how to leverage the characteristics of Q_i and the bit conditions on Y_{i-4} and Y_i to calculate $P(\mu\nu)$. The general case can be handled in the same way.

Example. For the given differential path in Table 16, we know that

$$\begin{aligned}
\Delta F &= ONX(Y'_{14}, Y'_{13}, (Y'_{12})^{\lll 10}) \boxminus ONX(Y_{14}, Y_{13}, Y^{\lll 10}_{12}) = \texttt{0xbffa20},\\
Y'^{\lll 10}_{11} &= Y^{\lll 10}_{11}[-26, 25, 22, 20, 11, -6, 4, 2],\ \Delta_{11} = Y'^{\lll 10}_{11} \boxminus Y^{\lll 10}_{11} = \texttt{0xfe5007d4},\\
Y'^{\lll 10}_{10} &= Y^{\lll 10}_{10}[31, -13, -6, 4, 3, 2, 1, 0],\ \Delta_{10} = Y'^{\lll 10}_{10} \boxminus Y^{\lll 10}_{10} = \texttt{0x7fffdfdf},\\
\Delta &= Y'_{15} - Y_{15} = \texttt{0xfbefffe0},\ \Delta m_3 = 0.
\end{aligned}$$

Therefore, Q_{15} has to satisfy the equation $(Q_{15} \boxplus \texttt{0x80bfd9ff})^{\lll 12} = Q^{\lll 12}_{15} \boxplus \texttt{0xfd9ff80c}$. According to the example in Sect. 3.2, the characteristics of Q_{15} which satisfies such an equation can be deduced and we display it in Table 4.

Let $a = Q^{\lll 12}_{15}$, $b = Y^{\lll 10}_{11}$, $d = Y_{15}$, since $Y_{15} = Y^{\lll 10}_{11} \boxplus Q^{\lll 12}_{15}$, we can obtain that $d = a \boxplus b$. In addition, we denote by c_i the carry from the(i-1)-th bit to the i-th bit when calculating $a \boxplus b$. Thus,

$$[d]_i = [a]_i \oplus [b]_i \oplus c_i,\ (c_0 = 0,\ 0 \le i \le 31).$$

Define as A_i the event that $[a]_i = 0$, as B_i the event that $[b]_i = 0$, as λ_i the event that $c_i = 0$, as D_i the event that $Y_{15,i} = 0$, as ν_{15} the event that $Y'_{15} - Y_{15} = \texttt{0xfbefffe0}$, as η_{15} the event that all the 7 conditions on Y_{15} hold. For a better understanding of our method, we display the calculation of Y_{15} in Table 5. Then, $P(\eta_{15}\nu_{15})$ can be calculated as follows:

Table 4. The characteristics of Q_{15}

i	χ_i (Characteristic)	Type	i	χ_i (Characteristic)	Type
1	$[Q_{15}]_{31} = 1$	Impossible	11	$[Q_{15}]_{19\sim11} = 000000101$	Possible
2	$[Q_{15}]_{31\sim23} = 011111111$	Impossible	12	$[Q_{15}]_{19\sim8} = 000000100111$	Possible
3	$[Q_{15}]_{31\sim22} = 0111111101$	Impossible	13	$[Q_{15}]_{19\sim7} = 0000001001101$	Possible
4	$[Q_{15}]_{19} = 1$	Possible	14	$[Q_{15}]_{19\sim6} = 00000010011001$	Possible
5	$[Q_{15}]_{19\sim18} = 01$	Possible	15	$[Q_{15}]_{19\sim5} = 000000100110001$	Possible
6	$[Q_{15}]_{19\sim17} = 001$	Possible	16	$[Q_{15}]_{19\sim4} = 0000001001100001$	Possible
7	$[Q_{15}]_{19\sim16} = 0001$	Possible	17	$[Q_{15}]_{19\sim3} = 00000010011000001$	Possible
8	$[Q_{15}]_{19\sim15} = 00001$	Possible	18	$[Q_{15}]_{19\sim2} = 000000100110000001$	Possible
9	$[Q_{15}]_{19\sim14} = 000001$	Possible	19	$[Q_{15}]_{19\sim1} = 0000001001100000001$	Possible
10	$[Q_{15}]_{19\sim12} = 00000011$	Possible	20	$[Q_{15}]_{19\sim0} = 00000010011000000001$	Possible

Table 5. Calculation of $Y_{15} = Y_{11}^{\lll 10} \boxplus Q_{15}^{\lll 12}$

$Q_{15}^{\lll 12}$	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20
$Y_{11}^{\lll 10}$	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22
Y_{15}	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$Q_{15}^{\lll 12}(a)$																																
$Y_{11}^{\lll 10}(b)$	-	1	-	-	1	1	0	-	-	0	-	0	-	-	1	0	0	0	0	1	0	-	0	-	-	1	-	0	1	0	-	-
$Y_{15}(d)$	-	-	-	-	-	1	-	-	-	1	-	1	0	-	-	-	-	-	-	-	1	-	-	-	-	-	1	-	-	1	-	-

$$
\begin{aligned}
P(\eta_{15}\nu_{15}) &= P(\lambda_{12}\eta_{15}\nu_{15}) + P(\overline{\lambda_{12}}\eta_{15}\nu_{15}),\\
P(\lambda_{12}\eta_{15}\nu_{15}) &= \Sigma_{i=4}^{20} P(\overline{D_{26}D_{22}D_{20}}D_{19}\chi_i \mid \lambda_{12}) \times \{P(\overline{D_{11}D_5D_2}\lambda_{12})\\
&\quad -\Sigma_{i=1}^{3}[P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_i) \times P(\chi_i)]\},\\
P(\overline{\lambda_{12}}\eta_{15}\nu_{15}) &= \Sigma_{i=4}^{20} P(\overline{D_{26}D_{22}D_{20}}D_{19}\chi_i \mid \overline{\lambda_{12}}) \times \{P(\overline{D_{11}D_5D_2\lambda_{12}})\\
&\quad -\Sigma_{i=1}^{3}[P(\overline{D_{11}D_5D_2\lambda_{12}} \mid \chi_i) \times P(\chi_i)]\}.
\end{aligned}
$$

However, according to the characteristics of Q_{15}, we know that $[Q_{15}]_{31}$ is always 0 if $Y'_{15} \boxminus Y_{15} =$ `0xfbefffe0`, which implies that $P(\overline{\lambda_{12}} \mid \nu_{15}) = 0$ and $P(\lambda_{12} \mid \nu_{15}) = 1$. Therefore, calculating $P(\lambda_{12}\eta_{15}\nu_{15})$ is enough. Take the calculation of $P(\overline{D_{11}D_5D_2}\lambda_{12}) - \Sigma_{i=1}^{3}[P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_i) \times P(\chi_i)]$ as an example. Firstly, we calculate $P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_3)$. As Table 6 shows, the calculation is detailed as below.

Table 6. Calculation of $P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_3)$

$Q_{15}^{\lll 12}$	31	30	29	28	27	26	25	24	23	22	21	20
$Y_{11}^{\lll 10}$	1	0	31	30	29	28	27	26	25	24	23	22
Y_{15}	11	10	9	8	7	6	5	4	3	2	1	0
$Q_{15}^{\lll 12}(a)$	0	1	1	1	1	1	1	1	0	1	-	-
$Y_{11}^{\lll 10}(b)$	0	-	0	-	-	1	-	0	1	0	-	-
$Y_{15}(d)$	1	-	-	-	-	-	1	-	-	1	-	-

$P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_3) = P(\overline{D_{11}D_5D_2} \mid \chi_3)$.

$P(\overline{D_{11}D_5D_2} \mid \chi_3) = P(\overline{\lambda_{11}D_5D_2} \mid \chi_3) = P(\overline{B_{10}}) \times P(\overline{D_5D_2} \mid \chi_3) + P(B_{10}) \times P(\overline{\lambda_{10}D_5D_2} \mid \chi_3)$

$= \frac{1}{2} \times P(\overline{D_5D_2} \mid \chi_3) + \frac{1}{2} \times P(\overline{\lambda_{10}D_5D_2} \mid \chi_3)$.

$P(\overline{D_5D_2} \mid \chi_3) = P(B_5) \times P(\lambda_5\overline{D_2} \mid \chi_3) + P(\overline{B_5}) \times P(\overline{\lambda_5D_2} \mid \chi_3)$

$= \frac{1}{2} \times P(\lambda_5\overline{D_2} \mid \chi_3) + \frac{1}{2} \times P(\overline{\lambda_5D_2} \mid \chi_3) = \frac{1}{2} \times P(\overline{D_2} \mid \chi_3)$.

$P(\overline{D_2} \mid \chi_3) = P(\lambda_2)$.

$P(\lambda_2) = P(A_1B_1) + [P(\overline{A_1}B_1) + P(A_1\overline{B_1})] \times P(\lambda_1) = \frac{1}{4} + \frac{1}{2} \times P(\lambda_1)$.

$P(\lambda_1) = P(A_1B_1) + P(\overline{A_1}B_1) + P(A_1\overline{B_1}) = \frac{3}{4}$.

$P(\overline{\lambda_{10}D_5D_2} \mid \chi_3) = P(\overline{\lambda_9D_5D_2} \mid \chi_3) = P(\overline{B_8}) \times P(\overline{D_5D_2} \mid \chi_3) + P(B_8) \times P(\overline{\lambda_8D_5D_2} \mid \chi_3)$

$= \frac{1}{2} \times P(\overline{D_5D_2} \mid \chi_3) + \frac{1}{2} \times P(\overline{\lambda_8D_5D_2} \mid \chi_3)$.

$P(\overline{\lambda_8D_5D_2} \mid \chi_3) = P(\overline{B_7}) \times P(\overline{D_5D_2} \mid \chi_3) + P(B_7) \times P(\overline{\lambda_7D_5D_2} \mid \chi_3)$

$= \frac{1}{2} \times P(\overline{D_5D_2} \mid \chi_3) + \frac{1}{2} \times P(\overline{\lambda_7D_5D_2} \mid \chi_3)$.

$P(\overline{\lambda_7D_5D_2} \mid \chi_3) = P(\overline{D_5D_2} \mid \chi_3)$.

Therefore, $P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_3) = \frac{5}{16}$. In the same way, we can obtain that $P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_2) = \frac{1}{4}$, $P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_1) = \frac{159}{1024}$ and $P(\overline{D_{11}D_5D_2}\lambda_{12}) = P(\overline{D_{11}D_5D_2}) = \frac{1}{8}$. Hence,

$$P(\overline{D_{11}D_5D_2}\lambda_{12}) - \Sigma_{i=1}^{3}[P(\overline{D_{11}D_5D_2}\lambda_{12} \mid \chi_i) \times P(\chi_i)]$$
$$= \frac{1}{8} - \frac{1}{2} \times \frac{159}{1024} - \frac{1}{2^9} \times \frac{1}{4} - \frac{1}{2^{10}} \times \frac{5}{16} \approx \frac{1}{16} = 2^{-4}.$$

Since

$$\Sigma_{i=4}^{20}P(\overline{D_{26}D_{22}D_{20}}D_{19}\chi_i \mid \lambda_{12}) = \Sigma_{i=4}^{20}[P(\overline{D_{26}D_{22}D_{20}}D_{19} \mid \chi_i\lambda_{12}) \times P(\chi_i \mid \lambda_{12})]$$
$$= \Sigma_{i=4}^{20}[P(\overline{D_{26}D_{22}D_{20}}D_{19} \mid \chi_i\lambda_{12}) \times P(\chi_i)],$$

and $P(\overline{D_{26}D_{22}D_{20}}D_{19} \mid \chi_i\lambda_{12})$ $(4 \leq i \leq 20)$ can be calculated in the same way as above, the value of $\Sigma_{i=4}^{20}P(\overline{D_{26}D_{22}D_{20}}D_{19}\chi_i \mid \lambda_{12})$ can be obtained. Thus, the probability of the step function can be calculated.

In summary, in order to theoretically calculate the step differential probability for step i, we should firstly deduce the characteristics of Q_i so that the modular difference can be correctly propagated. Then, for each characteristics of Q_i, the calculation of the probability that the bit conditions hold is changed to calculating the probability that $A + B = C$ where only part bits of A and B are fixed and some bits of C are restricted to fixed values. When all characteristics of Q_i are considered, the step differential probability can be obtained.

4 Solving the Equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$

When using the method proposed by Landelle and Peyrin to analyze RIPEMD-128 and RIPEMD-160 [5], an equation like $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ is always constructed. In order to reduce the time complexity of the merging phase, pre-computing the equation becomes a feasible way. However, in the previous analysis [5,11], the method of pre-computing the equation costs too much time and memory. In this section, we propose a method to reduce the time complexity and memory complexity. Based on *Property 1*, given a constant C_1, if there exists a solution to the equation $(T \boxplus C_1)^{\lll S_1} = T^{\lll S_1} \boxplus C_2$, then C_2 can only take the following four possible values:

1. $C_2 = (C_1 \boxminus 1)^{\lll S_1}$, and $[C_2]_{(S_1-1)\sim 0} \neq 0$.
2. $C_2 = (C_1 \boxplus 2^{32-S_1})^{\lll S_1}$, and $[C_1]_{(31-S_1)\sim 0} \neq 0$.
3. $C_2 = (C_1 \boxplus 2^{32-S_1} \boxminus 1)^{\lll S_1}$, and $[C_2]_{(S_1-1)\sim 0} \neq 0, [C_1]_{(31-S_1)\sim 0} \neq 0$.
4. $C_2 = C_1^{\lll S_1}$.

Therefore, given a constant C_1, we can compute and store the four possible values of C_2 based on the relationship between C_1 and C_2 as above. Then, for each value of C_2, we need to solve the equation $T^{\lll S_0} \boxplus C_0 = T^{\lll S_1} \boxplus C_2$. Let $C_3 = C_0 \boxminus C_2$, the equation becomes $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$. Therefore, we only need to pre-compute the equation $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$. Then, in order to obtain the solutions to the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$, we only need to guess four possible values of C_2. For each guessed value of C_2, the solutions to the equation $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$ can be quickly obtained. For the obtained solution T, we have to verify whether it satisfies the equation $(T \boxplus C_1)^{\lll S_1} = T^{\lll S_1} \boxplus C_2$ since T satisfies it with probability. Pre-computing the equation $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$ only costs 2^{32} time and 2^{32} memory, which is much smaller.

The expectation of the number of the solution to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ also has an influence on the time complexity of the merging phase. Since it is not mentioned in the previous analysis, it is necessary to give a theoretical value. Consider the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$. Once we fix one constant, supposing that is C_0, and then exhaust all the 2^{32} possible values of T, the corresponding C_1 can be obtained. Since more than one value of T might correspond to the same C_1, one value of C_1 will correspond to more than one value of T if C_0 is fixed. We show it in Table 7.

Table 7. Number of the solutions

T	0	...	i	...	j	...	`0xffffffff`
C_1	x	...	x	...	x	...	y

C_1	0	...	i	...	j	...	k	...	`0xffffffff`
T		...	*NULL*	...	$T_{i_2}, T_{i_3}, T_{i_4}$	...	$T_{i_5}, T_{i_6}, T_{i_7}, T_{i_8}$	...	

When C_0 is fixed and C_1 is random, we denote by ε the number of the solutions, and denote by p_i the probability of that there are i solutions to the equation. In addition, we denote by N_i the number of C_1 which corresponds to i solutions to the equation. Suppose there are at most n solutions to the equation. Then, we can deduce that

$$N_1 + 2N_2 + \ldots + nN_n = 2^{32},$$
$$p_i = \frac{N_i}{2^{32}},$$
$$E(\varepsilon) = p_1 + 2p_2 + \ldots + np_n = \frac{N_1 + 2N_2 + \ldots + nN_n}{2^{32}} = 1.$$

Therefore, the number of expected solutions to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ is 1. In the same way, we can obtain that the number of expected solutions to $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$ is also 1.

In conclusion, given many pairs of (C_0, C_1), we can calculate the four corresponding possible values of C_2 at first. Since the number of expected solutions to $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$ is 1, we will obtain four possible solutions to $T^{\lll S_0} \boxplus C_3 = T^{\lll S_1}$ on average for the four values of C_2. However, we need to further check whether the four solutions T satisfy $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$. Since the expectation of the number of the solution to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ is 1, we will obtain one solution to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ on average. Therefore, when solving the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$, only four times of check is enough on average, which is very quick. Therefore, the time complexity of solving the equation is 2^2.

5 Collision Attack on the First 30-Step RIPEMD-160

By constructing a 30-step differential path, where the left branch is sparse and the right branch is controlled as sparse as possible, applying the message modification techniques proposed by Wang [20] to the right branch while the left branch remains probabilistic, it is possible to mount a collision attack on 30-step RIPEMD-160 with probability 2^{-67}. The 30-step differential path is shown in Table 8. Using the single-step modification and multi-step modification in [20], the bit conditions on the internal states can be satisfied. As mentioned before, the differential path holds only when both the modular difference of the internal states and the bit conditions hold, which is different from MD4. However, the message modification techniques can't be directly used to ensure the modular difference of the internal states holds. Moreover, the probability that the modular difference of the internal states holds has a great effect on the phase of the message modification, the reason for this will be discussed later. Therefore, how to have the modular difference of the internal states hold when using the message modification becomes an urgent problem to be solved. According to the previous part to calculate the step differential probability, we can change such a problem into how to ensure Q_i satisfies its corresponding equation so that ΔY_i holds when using the message modification.

Table 8. 30-step differential path, where $m'_{15} = m_{15} \boxplus 2^{24}$, and $\Delta m_i = 0$ $(0 \leqslant i \leqslant 14)$. Note that the symbol n represents that a bit changes to 1 from 0, u represents that a bit changes to 0 from 1, and - represents that the bit value is free.

	X_i	$\pi_1(i)$		Y_i	$\pi_2(i)$
-4	--------------------------------		-4	--------------------------------	
-3	--------------------------------		-3	--------------------------------	
-2	--------------------------------		-2	--------------------------------	
-1	--------------------------------		-1	--------------------------------	
00	--------------------------------	00	00	--------------------------------	05
01	--------------------------------	01	01	--------------------------------	14
02	--------------------------------	02	02	--------------------------------	07
03	--------------------------------	03	03	--------------------------------	00
04	--------------------------------	04	04	--------------------------------	09
05	--------------------------------	05	05	--------------------------------	02
06	--------------------------------	06	06	--------------------------------	11
07	--------------------------------	07	07	--------------------------------	04
08	--------------------------------	08	08	--------------------------------	13
09	--------------------------------	09	09	-----1-1-1----------------------	06
10	--------------------------------	10	10	----000000-1--1---0000--1-001010	15
11	--------------------------------	11	11	-0--0---0000110110010000000nuuuu	08
12	--------------------------------	12	12	nuuuuuuuuuuuuuuuu0n0n00----01100	01
13	--------------------------------	13	13	0unn1uu-111-1-1--nuunn11011011un	10
14	--------------------------------	14	14	-100001111----1-10nu10101-nu1-11	03
15	--------------------------------	15	15	00---01111-0u-u-101000-u----0-01	12
16	-------------------------------n	07	16	111-n1uu000n1n--0001n----nuuuuuu	06
17	-------------------------------0	04	17	1u1-1--un--0111-00u10unnn-nnn01-	11
18	---------------------1---------1	13	18	01------0n-011--1n0000----0-00-1	03
19	---------------------0----------	01	19	1u------1--100--010----------1-1	07
20	---------------------n----------	10	20	-0--------1---------0nu11---11-0	00
21	---------------------0----------	06	21	-1-----1011-----11111-101-------	13
22	-----------1---------1----------	15	22	u-----001-u-----------1u------00	05
23	n----------0--------------------	03	23	1--------------0-----01-------n-	10
24	0----------n--------------------	12	24	1--------------1----0-1-------00	14
25	1----------0----------1---------	00	25	1----n-----0--------1---------01	15
26	-1---------1----------0---------	09	26	-----------0--------unn---------	08
27	-0--------------------n---------	05	27	-u------------------------------	12
28	-n--------------------0---------	02	28	--------------------------------	04
29	-0----------1-------------------	14	29	--------------------------------	09
30	--------------------------------	11	30	--------------------------------	01

Other Conditions
$Y_{11,31} \bigvee \neg Y_{10,21} = 1$, $Y_{11,29} \bigvee \neg Y_{10,19} = 1$, $Y_{11,28} \bigvee \neg Y_{10,18} = 1$, $Y_{11,26} \bigvee \neg Y_{10,16} = 1$, $Y_{11,25} \bigvee \neg Y_{10,15} = 1$, $Y_{11,24} \bigvee \neg Y_{10,14} = 1$.
$Y_{14,21} = 1$, $Y_{14,20} = 1$, $Y_{14,9} = 1$ (We use the three conditions); Or $Y_{15,21} = 1$, $Y_{14,21} = 0$, $Y_{14,20} = 0$, $Y_{14,19} = 0$.
$Y_{15,6} = 1$, $Y_{14,6} = 0$, $Y_{15,5} = 1$; Or $Y_{14,6} = 1$, $Y_{15,5} = 0$ (We use the two conditions).
$Y_{15,29} = 0$, $Y_{15,28} = 0$, $Y_{15,27} = 1$.
$Y_{18,28} = Y_{17,28}$, $Y_{18,21} = Y_{17,21}$, $Y_{18,16} = Y_{17,16}$.
$Y_{19,17} = Y_{18,17}$, $Y_{19,8} = Y_{18,8}$, $Y_{19,1} = Y_{18,1}$.
$Y_{20,24} = Y_{19,24}$.
$Y_{22,19} = Y_{21,19}$, $Y_{22,20} = Y_{21,20}$.
$Y_{28,19} = Y_{27,19}$, $Y_{28,20} = Y_{27,20}$, $Y_{28,21} = Y_{27,21}$.
$X_{15,0} = X_{14,22}$.
$X_{22,31} = X_{21,21}$.

5.1 Deducing Extra Bit Conditions to Control the Characteristics of Q_i

Given a differential path, both the bit conditions on the internal states and the equations that all Q_i have to satisfy are fixed. The differential path holds only when all these bit conditions hold and all Q_i satisfy their corresponding equations. Although the message modification techniques proposed by Wang can be used to ensure the bit conditions on the internal states hold, it can't be directly used to ensure Q_i satisfies its corresponding equation. However, if we can add some extra bit conditions on Y_i and Y_{i-4} to ensure Q_i always satisfies its corresponding equation, the influence of Q_i can be eliminated. Then, the message modification ensures that both the bit conditions and the modular differences of the internal state hold at the same time. Taking Q_{13} as an example, we show how to deduce the extra bit conditions on Y_{13} and Y_9.

Based on the 30-step differential path in Table 8, we can obtain that Q_{13} has to satisfy the equation $(Q_{13} \boxplus \texttt{0x6ffba800})^{\lll 14} = Q_{13}^{\lll 14} \boxplus \texttt{0xea001bff}$ so that the modular difference ΔY_{13} holds, from which we can deduce the characteristics of Q_{13} as described before. We only choose two possible characteristics of Q_{13}, which are $[Q_{13}]_{31} = 0$ and $[Q_{13}]_{17} = 1$. By applying the single-step message modification, all the bit conditions on Y_{13} and Y_9 can be satisfied, which means that some bits of them are fixed. Considering the relationship between Y_{13} and Y_9:

$$Q_{13}^{\lll 14} = Y_{13} \boxminus Y_9^{\lll 10},$$

our goal is to ensure the two bit conditions on Q_{13} are satisfied under the condition that some bits of Y_{13} and Y_9 are already fixed. We show the calculation of $Q_{13}^{\lll 14} = Y_{13} \boxminus Y_9^{\lll 10}$ in Table 9, which will help understand how to accurately deduce the extra bit conditions.

Table 9. The calculation of $Q_{13}^{\lll 14} = Y_{13} \boxminus Y_9^{\lll 10}$

Y_{13}	0 1 0 0	1 u u -	1 1 1 -	1 - 1 -	- n u u	n n 1 1	0 1 1 0	1 1 u n
$Y_9^{\lll 10}$	1 0 - -	- - - -	- - - -	- - - -	- - 1 0	- - - -	- - - 1	- 1 - 1
$Q_{13}^{\lll 14}$	1 - - -	- - - -	- - - -	- - - -	- - 0 -	- - - -	- - - -	- - - -

If we impose four bit conditions on Y_9, which are $Y_{9,2} = 0$, $Y_{9,3} = 1$, $Y_{9,20} = 0$, $Y_{9,21} = 1$, the two bit conditions on Q_{13} will hold with probability 1. In other words, if all the bit conditions (including the extra conditions) on Y_9 and Y_{13} hold, the equation $(Q_{13} \boxplus \texttt{0x6ffba800})^{\lll 14} = Q_{13}^{\lll 14} \boxplus \texttt{0xea001bff}$ will always hold. Therefore, by adding four extra conditions on Y_9, the message modification can ensure both the bit conditions on Y_{13} and the modular difference ΔY_{13} hold.

Sometimes, however, adding many extra conditions costs too much. Therefore, for some special cases, we use a dynamic way to add fewer conditions to ensure that Q_i satisfies its corresponding equation with probability 1 or close to 1. For example, in order to ensure that the modular difference ΔY_{23} holds, Q_{23} has to satisfy the equation $(Q_{23} \boxplus \texttt{0x81000001})^{\lll 9} = Q_{23}^{\lll 9} \boxplus \texttt{0x102}$, from which we can deduce the characteristics of Q_{23}. Then, we choose one possible characteristic, which is $[Q_{23}]_{31} = 1$. In this way, Q_{23} satisfies its corresponding equation with probability $1 - 2^{-23} \approx 1$. By considering the calculation of $Q_{23}^{\lll 9} = Y_{23} \boxminus Y_{19}^{\lll 10}$ as shown in Table 10, we describe how to dynamically determine the bit conditions on Y_{23}.

Table 10. The calculation of $Q_{23}^{\lll 9} = Y_{23} \boxminus Y_{19}^{\lll 10}$

Y_{23}	1 - - -	- - - -	- - - -	- - - 0	- - - -	- 0 1 -	- - - -	- - n -
$Y_{19}^{\lll 10}$	1 u - -	- - - -	1 - - 1	0 0 - -	0 1 0 -	- - - -	- - - -	- 1 - 1
$Q_{23}^{\lll 9}$	- - - -	- - - -	- - - -	- - - -	- - - -	- - - 1	- - - -	- - - -

According to the multi-step message modification [20], we should deal from lower bits to higher bits to correct Y_{23}. Therefore, we compare $[Y_{23}]_{7\sim 0}$ with $[Y_{19}^{\lll 10}]_{7\sim 0}$ only when $Y_{23,1}$ has been corrected. For different relationships between them, we should determine the bit conditions differently. By dynamically determine the conditions on Y_{23} in this way, we can ensure Q_{23} satisfies its corresponding equation with probability close to 1 by applying the message modification to correct $Y_{23,8}$.

1. If $[Y_{23}]_{7\sim 0} \geq [Y_{19}^{\lll 10}]_{7\sim 0}$, we add a condition $Y_{23,8} \bigoplus Y_{19,30} = 1$.
2. If $[Y_{23}]_{7\sim 0} < [Y_{19}^{\lll 10}]_{7\sim 0}$, we add a condition $Y_{23,8} \bigoplus Y_{19,30} = 0$.

As described above, we can deduce many extra bit conditions on the internal states, and they are displayed in Table 11. Then we can take these newly added bit conditions into consideration when applying the message modification techniques. In this way, both the bit conditions and the modular difference of the internal states can be satisfied at the same time.

Table 11. Equations of Q_i for the 30-step differential path and extra conditions to control the equations

Equation: $(Q_i \boxplus in)^{\lll shift} = Q_i^{\lll shift} \boxplus out$				
i	shift	in	out	Extra conditions
11	8	0x1000000	0x1	$Y_{7,24} = 1$
12	11	0x15	0xa800	$Y_{8,21} = 0,\ Y_{8,19} = 0$
13	14	0x6ffba800	0xea001bff	$Y_{9,3} = 1,\ Y_{9,2} = 0,\ Y_{9,21} = 1,\ Y_{9,20} = 0$
14	14	0x40400001	0x1010	$Y_{10,31} = 0$
15	12	0xafffff5f	0xfff5fb00	$Y_{15,9} = 0,\ Y_{11,31} = 1$
16	6	0x9d020	0x2740800	
17	9	0x85f87f2	0xbf0fe410	$Y_{13,20} = 1,\ Y_{13,18} = 0,\ Y_{17,28} = 0,\ Y_{17,26} = 1,\ Y_{13,16} = 0$
18	7	0x0	0x0	
19	15	0xffffd008	0xe8040000	$Y_{15,21} = 0$
20	7	0xd75fbffc	0xafdffdec	
21	12	0x10200813	0x812102	$Y_{21,6} = 1,\ Y_{17,28} = 0,\ Y_{21,10} = Y_{17,0}$
22	8	0xff7edffe	0x7edffeff	$Y_{22,30} = 1,\ Y_{18,21} = 1,\ Y_{22,2} = Y_{18,24},\ Y_{22,3} = Y_{18,25},$ $Y_{22,4} = Y_{18,26},\ Y_{22,5} = Y_{18,27},\ Y_{22,6} = Y_{18,28},\ Y_{22,7} = Y_{18,29}$
23	9	0x81000001	0x102	If $[Y_{23}]_{7\sim 0} \geq [Y_{19}^{\lll 10}]_{7\sim 0}$, then $Y_{23,8} \bigoplus Y_{19,30} = 1$ If $[Y_{23}]_{7\sim 0} < [Y_{19}^{\lll 10}]_{7\sim 0}$, then $Y_{23,8} \bigoplus Y_{19,30} = 0$
24	11	0xffffff00	0xfff80000	
25	7	0x80000	0x4000000	
26	7	0x1000800	0x80040000	
27	12	0x7ffc0000	0xbffff800	
28	7	0x0	0x0	
29	6	0xc0000000	0xfffffff0	
30	15	0x10	0x80000	

5.2 Multi-step Modification for RIPEMD-160

After obtaining the newly added bit conditions, we need to apply the message modification techniques to correct the bits of the internal states. Since the single-step modification is relatively simple, we refer the interested readers to [20] for more details. The following is an example to correct the three bit conditions on Y_1 by single-step modification. For the first round, we can correct the bit conditions on the internal states in this similar way.

$$\begin{aligned} Y_1 &\longleftarrow Y_1 \oplus (\overline{Y_{1,3}} \lll 3) \oplus (\overline{Y_{1,14}} \lll 14) \oplus (\overline{Y_{1,29}} \lll 29). \\ m_5 &\longleftarrow (Y_1 \boxminus Y_{-3}^{\lll 10})^{\ggg 8} \boxminus (Y_{-4}^{\lll 10} \boxplus ONX(Y_0, Y_{-1}, Y_{-2}^{\lll 10}) \boxplus K_0^r). \end{aligned}$$

For the internal states after the first round, the multi-step modification should be applied. However, the step function of RIPEMD-160 is no longer a T-function. Therefore, the multi-step modification for RIPEMD-160 is slightly different from that for MD4 [20]. We take correcting $Y_{17,4}$, $Y_{17,3}$ and $Y_{23,16}$ as three examples to show three types of multi-step modification for RIPEMD-160.

Table 12. Message modification for correcting $Y_{17,4}$

			Modify m	New internal state	Q
Y_9	m_{13}	7	$m_{13} \longleftarrow m_{13} \boxplus (Q_9^{\lll 7} \boxplus 2^{27})^{\ggg 7} \boxminus Q_9)$	$Y_9^{new} = Y_9[27]$	Q_9 is changed.
Y_{10}	m_6	7	$m_6 \longleftarrow (Y_{10} \boxminus Y_6^{\lll 10})^{\ggg 7} \boxminus Y_5^{\lll 10} \boxminus ONX(Y_9^{new}, Y_8, Y_7^{\lll 10}) \boxminus K_0^r$	Y_{10}	Q_{10} stays the same.
Y_{11}	m_{15}	8	$m_{15} \longleftarrow (Y_{11} \boxminus Y_7^{\lll 10})^{\ggg 8} \boxminus Y_6^{\lll 10} \boxminus ONX(Y_{10}, Y_9^{new}, Y_8^{\lll 10}) \boxminus K_0^r$	Y_{11}	Q_{11} stays the same.
Y_{12}	m_8	11	$m_8 \longleftarrow (Y_{12} \boxminus Y_8^{\lll 10})^{\ggg 11} \boxminus Y_7^{\lll 10} \boxminus ONX(Y_{11}, Y_{10}, Y_9^{new \lll 10}) \boxminus K_0^r$	Y_{12}	Q_{12} stays the same.
Y_{13}	m_1	14	$m_1 \longleftarrow (Y_{13} \boxminus Y_9^{new \lll 10})^{\ggg 14} \boxminus Y_8^{\lll 10} \boxminus ONX(Y_{12}, Y_{11}, Y_{10}^{\lll 10}) \boxminus K_0^r$	Y_{13}	Q_{13} is changed.
Y_{14}	m_{10}	14	$m_{10} \longleftarrow (Y_{14} \boxminus Y_{10}^{\lll 10})^{\ggg 14} \boxminus Y_9^{new \lll 10} \boxminus ONX(Y_{13}, Y_{12}, Y_{11}^{\lll 10}) \boxminus K_0^r$	Y_{14}	Q_{14} stays the same.

In order to correct $Y_{17,4}$, we can change the 27th bit of m_6. Therefore, we can change the 27th bit of Y_9 by changing the value of m_{13}. Then, modify m_6, m_{15}, m_8, m_1, m_{10} to have Y_i $(10 \leq i \leq 14)$ remaining the same. In this way, $Y_{17,4}$ can be corrected. According to Table 12, we can find that Q_9 and Q_{13} are changed during the phase of message modification. Since there is no constraints on Q_9, it doesn't matter if Q_9 is changed. However, Q_{13} has to satisfy the equation $(Q_{13} \boxplus \texttt{0x6ffba800})^{\lll 14} = Q_{13}^{\lll 14} \boxplus \texttt{0xea001bff}$ so that the modular difference ΔY_{13} holds. Thus, we have to consider the influence of its change. As introduced in the previous part, we have added some extra conditions on Y_9 to ensure Q_{13} will always satisfy this equation under the condition that all bit conditions on Y_9 and Y_{13} hold. Although Y_9 is changed when correcting $Y_{17,4}$, it won't have an influence on the conditions added to control the characteristics of Q_{13}, which means that Q_{13} still satisfies its corresponding equation even though it is changed. The main reason is that we have controlled the characteristics of Q_{13} by the newly added bit conditions and such a Q_{13} will always satisfy its corresponding equation. If we don't pre-deduce the extra bit conditions to control the characteristics of Q_{13}, the equation $(Q_{13} \boxplus \texttt{0x6ffba800})^{\lll 14} = Q_{13}^{\lll 14} \boxplus \texttt{0xea001bff}$ may not hold any more since Q_{13} has been changed. In other word, $Y_{17,4}$ may be probabilistically corrected. And the probability is equal to the probability that the equation

Table 13. Message modification for correcting $Y_{23,16}$

			Modify m	New internal state	Extra Conditions
Y_1	m_5	8	$m_5 \longleftarrow m_5 \boxplus 2^7$	$Y_1^{new} = Y_1[15]$	$(Q_1 \boxplus 2^7)^{\lll 8} = Q_1^{\lll 8} \boxplus 2^{15}$.
Y_2	m_{14}	9	$m_{14} = (Y_2 \boxminus Y_{-2}^{\lll 10})^{\ggg 9} \boxminus Y_{-3}^{\lll 10} \boxminus ONX(Y_1^{new}, Y_0, Y_{-1}^{\lll 10}) \boxminus K_0^r$	Y_2	
Y_3	m_7	9		Y_3	$Y_{0,5} = 0$.
Y_4	m_0	11		Y_4	$Y_{2,25} = 1$.
Y_5	m_9	13	$m_9 \longleftarrow (Y_5 \boxminus Y_1^{new \lll 10})^{\ggg 13} \boxminus Y_0^{\lll 10} \boxminus ONX(Y_4, Y_3, Y_2^{\lll 10}) \boxminus K_0^r$	Y_5	
Y_6	m_2	15	$m_2 \longleftarrow (Y_6 \boxminus Y_2^{\lll 10})^{\ggg 15} \boxminus Y_1^{new \lll 10} \boxminus ONX(Y_5, Y_4, Y_3^{\lll 10}) \boxminus K_0^r$	Y_6	

Table 14. Message modification for correcting $Y_{17,3}$

			Modify m	New internal state	Q
Y_6	m_2	15	$Y_6^{new} \longleftarrow [Y_{10} \boxminus (Y_5^{\lll 10} \boxplus ONX(Y_9, Y_8, Y_7^{\lll 10}) \boxplus m_6 \boxplus 2^{26} \boxplus K_0^r)^{\lll 7}]^{\ggg 10}$ $m_2 \longleftarrow (Y_6^{new} \boxminus Y_2^{\lll 10})^{\ggg 15} \boxminus Y_1^{\lll 10} \boxminus ONX(Y_5, Y_4, Y_3^{\lll 10}) \boxminus K_0^r$	Y_6^{new}	Q_6 is changed.
Y_7	m_{11}	15	$m_{11} \longleftarrow (Y_7 \boxminus Y_3^{\lll 10})^{\ggg 15} \boxminus Y_2^{\lll 10} \boxminus ONX(Y_6^{new}, Y_5, Y_4^{\lll 10}) \boxminus K_0^r$	Y_7	Q_7 stays the same.
Y_8	m_4	5	$m_4 \longleftarrow (Y_8 \boxminus Y_4^{\lll 10})^{\ggg 5} \boxminus Y_3^{\lll 10} \boxminus ONX(Y_7, Y_6^{new}, Y_5^{\lll 10}) \boxminus K_0^r$	Y_8	Q_8 stays the same.
Y_9	m_{13}	7	$m_{13} \longleftarrow (Y_9 \boxminus Y_5^{\lll 10})^{\ggg 7} \boxminus Y_4^{\lll 10} \boxminus ONX(Y_8, Y_7, Y_6^{new \lll 10}) \boxminus K_0^r$	Y_9	Q_9 stays the same.
Y_{10}	m_6	7	$m_6 \longleftarrow m_6 \boxplus 2^{26}$	Y_{10}	Q_{10} is changed.
Y_{11}	m_{15}	8	$m_{15} \longleftarrow (Y_{11} \boxminus Y_7^{\lll 10})^{\ggg 8} \boxminus Y_6^{new \lll 10} \boxminus ONX(Y_{10}, Y_9, Y_8^{\lll 10}) \boxminus K_0^r$	Y_{11}	Q_{11} stays the same.

$(Q_{13} \boxplus \texttt{0x6ffba800})^{\lll 14} = Q_{13}^{\lll 14} \boxplus \texttt{0xea001bff}$ holds, which is about $2^{-0.5}$. Moreover, if we correct n bits of Y_{17} by using the strategy as Table 12 displays and don't pre-deduce the extra bit conditions, the probability that they are right corrected is about $2^{-0.5n}$, which will have a great effect on the probability to mount the collision attack on 30-step RIPEMD-160. Therefore, it is significant to pre-deduce the extra bit conditions to control the characteristics of Q_i, which will improve the time complexity of the message modification.

In order to correct $Y_{23,16}$, we can change the 7th bit of m_5. As displayed in Table 13, by modifying some message words and adding some extra conditions on the internal states, $Y_{23,16}$ can be corrected. For the strategy in Table 13, $Y_{23,16}$ can be corrected with probability that the equation $(Q_1 \boxplus 2^7)^{\lll 8} = Q_1^{\lll 8} \boxplus 2^{15}$ holds, which is $1 - 2^{-17} \approx 1$. Therefore, we can ignore the influence of this equation. Sometimes, however, such an equation holds with a low probability, which is bad for the correcting. In order to eliminate the influence, we can use the same idea in Sect. 5.1 to pre-deduce some extra bit conditions to control the characteristics of Q_i so that Q_i will satisfy such an equation.

In order to correct $Y_{17,3}$, we can change the 26th bit of m_6. Firstly, we compute a new value of Y_6 so that Y_{10} can stay the same only by adding 2^{26} to m_6. Then, a new value of m_2 can be obtained. To have Y_i $(7 \le i \le 11)$ remaining the same, m_{11}, m_4, m_{13}, m_6, m_{15} should be accordingly modified. As for strategy displayed in Table 14, it is because there is no condition on Y_6 that we can choose such a method to correct $Y_{17,3}$. Since there is no condition on Y_3 either, $Y_{18,31}$ can be corrected by using the similar strategy.

The multi-step message modification is summarized in Table 15. In this table, we also display some extra bit conditions to control the characteristics of Q_1 and Q_4 so that the newly added bit conditions on them for message modification can be satisfied. Although some of the equations of Q_1 and Q_4 remain uncontrolled, they will hold with probability close to 1.

Table 15. Summarization of the multi-step modification for Y_i $(17 \leq i \leq 23)$

Chaining variables	Bits to be corrected (i)	Chaining variables used	Extra Conditions
Y_{17}	1,2,12,13,14,15,23,24,30,31,21	Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}	$Y_5[i-19]$.
Y_{17}	4,5,7,8,9,10,17,18,19,20,26,27,28	Y_9, Y_{10}, Y_{11}, Y_{12}, Y_{13}, Y_{14}	$Y_9[i-9]$.
Y_{17}	11,29	Y_8, Y_9, Y_{10}, Y_{11}, Y_{12}, Y_{13}	$Y_8[i-9]$, $Y_{7,i-19}=1$.
Y_{17}	3	Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{11}	
Y_{18}	2,3,5,11,12,13,14,15,18,19,20,28,30	Y_2, Y_3, Y_4, Y_5, Y_6, Y_7	$Y_2[i-23]$.
Y_{18}	0,10,16,21,22,23	Y_4, Y_5, Y_6, Y_7, Y_8, Y_9	$Y_4[i-23]$, $Y_{5,i-13}=0$.
Y_{18}	31	Y_3, Y_4, Y_5, Y_6, Y_7, Y_8	
Y_{19}	19	Y_{15}, Y_{16}, Y_{17}, Y_{18}	$Y_{15}[16]$, $Y_{14,6}=1$, $Y_{16,26}=Y_{17,26}$.
Y_{20}	0,2,3,7,8,9,10,11,21,24,30	Y_1, Y_2, Y_3, Y_4, Y_5, Y_6	$Y_1[i-7]$, $Y_{0,i-17}=1$.
Y_{21}	7,8,9,13,15,22,23,24,30	Y_4, Y_5, Y_6, Y_7, Y_8, Y_9	$Y_4[i-1]$, $Y_{4,28}=1$, $Y_{4,27}=1$, $Y_{4,26}=1$, $Y_{0,19}=0$, $Y_{0,16}=0$,$Y_{4,5}=1$, $Y_{0,28}=0$, $Y_{0,27}=0$.
Y_{21}	6,10,11,12,14,21	Y_1, Y_2, Y_3, Y_4, Y_5, Y_6	$Y_1[i-22]$, $Y_{0,i}=0$, $Y_{2,i-12}=0$.
Y_{22}	0,1,2,3,4,5,6,7,8,9, 19,20,21,23,24,25,30,31	Y_8, Y_9, Y_{10}, Y_{11}, Y_{12}, Y_{13}	$Y_8[i-8]$, $Y_{7,i-18}=0$.
Y_{23}	8,9,10,16,31	Y_1, Y_2, Y_3, Y_4, Y_5, Y_6	$Y_1[i-1]$, $Y_{0,i-11}=0$, $Y_{2,i+9}=1$, $Y_{1,29}=1$, $Y_{-3,20}=0$, $Y_{-3,19}=0$. $Y_{1,6}=0$, $Y_{-3,29}=1$, $Y_{-3,28}=1$.
$Y_{4,28}=1$, $Y_{4,27}=1$, $Y_{4,26}=1$, $Y_{0,19}=0$, $Y_{0,16}=0$ are used to control: $(Q_4 \boxplus 2^{18})^{\lll 11} = Q_4^{\lll 11} \boxplus 2^{29}$.			
$Y_{4,5}=1$, $Y_{0,28}=0$, $Y_{0,27}=0$ are used to control: $(Q_4 \boxplus 2^{28})^{\lll 11} = Q_4^{\lll 11} \boxplus 2^7$ and $(Q_4 \boxplus 2^{29})^{\lll 11} = Q_4^{\lll 11} \boxplus 2^8$.			
$Y_{1,29}=1$, $Y_{-3,20}=0$, $Y_{-3,19}=0$ are used to control: $(Q_1 \boxplus 2^{22})^{\lll 8} = Q_1^{\lll 8} \boxplus 2^{30}$.			
$Y_{1,6}=0$, $Y_{-3,29}=1$, $Y_{-3,28}=1$ are used to control: $(Q_1 \boxplus 2^{31})^{\lll 8} = Q_1^{\lll 8} \boxplus 2^7$.			

5.3 Complexity Evaluation

For the left branch, we don't apply any message modification techniques to it. By randomly generating message words, we test the probability that the left branch holds. According to our experiments, the probability is about 2^{-29}.

For the right branch, we can use the message modification techniques to correct the bits of Y_i $(17 \leq i \leq 23)$. However, we can't find a way to correct all the bits of them, thus leaving 14 bit conditions remaining uncontrolled, i.e., 13 bits of Y_{19} and 1 bit of Y_{23}. Besides, to ensure Q_{20} can satisfy its corresponding equation with probability 1, some extra bit conditions on Y_{20} and Y_{16} should be added. However, it is difficult to have all these newly added bit conditions hold by using the message modification techniques, which will cause a lower probability. Therefore, we leave Q_{20} holding with probability about 2^{-1}. For Q_i $(11 \leq i \leq 23,\ i \neq 20)$, by correcting the newly added extra bit conditions, they will satisfy their corresponding equations with probability 1 or close to 1.

For Y_i $(24 \leq i \leq 30)$, since it is difficult to correct the 20 bit conditions on them, we leave them holding probabilistically. In addition, Q_i $(24 \leq i \leq 30)$ satisfy their corresponding equations with probability about 2^{-3}. Therefore, the right branch holds with probability about $2^{-14-1-20-3} = 2^{-38}$.

When applying the message modification techniques, we add 26 bit conditions on Y_0 and 4 bit conditions on Y_{-3}. Therefore, we need to use two message blocks (M_1, M_2) to mount the 30-step collision attack. M_1 is used to generate such a hash value that the bit conditions on Y_0 and Y_{-3} have been satisfied when compressing M_2, which costs $2^{26+4} = 2^{30}$ time. In conclusion, the 30-step collision attack succeeds with probability of about $2^{-29-38} = 2^{-67}$, and the time complexity is about $2^{67} + 2^{30} \approx 2^{67}$. The implementation of this attack is available at https://github.com/Crypt-CNS/RIPEMD160-30Step.git.

6 Improved Semi-Free-Start Collision Attack

6.1 36-Step Semi-Free-Start Collision Path

Mendel *et al.* [11] improved the techniques in [7,10], and used the improved algorithm to find two differential paths of RIPEMD-160. One is a 48-step semi-free-start collision path, the other is a 36-step semi-free-start collision path. Since we focus on the semi-free-start collision attack on the first 36-step RIPEMD-160, we only introduce the 36-step semi-free-start collision path. The differential path is displayed in Table 16. In order to have a full understanding of our improvement, it is necessary to briefly introduce the method proposed by Landelle and Peyrin [5].

The main idea of the method can be divided into three steps. Firstly, the attacker chooses the internal states in both branches and fixes some message words to ensure the non-linear parts. This step is called *find a starting point*.

Table 16. 36-step differential path, where $m'_7 = m_7 \boxplus 2^4 \boxplus 2^{15} \boxplus 2^{30}$, and $\Delta m_i = 0$ $(i \neq 7, 0 \leqslant i \leqslant 15)$. Note that the symbol n represents that a bit changes to 1 from 0, u represents that a bit changes to 0 from 1, and - represents that the bit value is free.

X_i		$\pi_1(i)$	Y_i		$\pi_2(i)$
-4	--------------------------------		-4	--------------------------------	
-3	--------------------------------		-3	--------------------------------	
-2	--------------------------------		-2	--------------------------------	
-1	--------------------------------		-1	--------------------------------	
00	--------------------------------	00	00	--------------------------------	05
01	--------------------------------	01	01	--1-------------1-----------1---	14
02	--------------------------------	02	02	--0---10---------10-----0---0---	07
03	--------------------------------	03	03	------1n---------1n----0n---0---	00
04	--------------------------------	04	04	100----n0-11-n0--------110--n0--	09
05	--------------------------------	05	05	n1-1---00----00n0-1---000---1101	02
06	--------------------------------	06	06	n--1000111110un---uuuuuuuu11-1u-	11
07	--------------------------------	07	07	uuu00un-n1u011nn000001101-01-n00	04
08	------n-uuuuuuuuuuu-nuuu-uuuuuuu	08	08	11u0uu--1-u1u0n1nn-nu-100---000u	13
09	--uun-nn-n---nnunnuu-------n--nn	09	09	0-uu011--01-u00011000n-nn01--111	06
10	-----n--unuun-u-u-----nn----u---	10	10	--1u1nnnnnn0110010-0-0-00100u1-0	15
11	--n---nuuu--nu--un-n------------	11	11	0--u-n1n---1--1un--n-n--100001n-	08
12	-----u----n-nnnnnnnnnnnnnnn-----	12	12	10-110----n0---0u1--1----0-111-n	01
13	--------------------------------	13	13	1u--0n-u-----------1-1u--10---1-	10
14	--------------------------------	14	14	----1----n-00--0----n111--n1-0--	03
15	--------------------------------	15	15	-----u---1-u0-------1-----u--1--	12
16	--------------------------------	07	16	---------------------------0----	06
17	--------------------------------	04	17	---------------------------1----	11
18	--------------------------------	13	18	--------------------------------	03
19	--------------------------------	01	19	--------------------------------	07
20	--------------------------------	10	20	--------------------------------	00
21	--------------------------------	06	21	--------------------------------	13
22	--------------------------------	15	22	--------------------------------	05
23	--------------------------------	03	23	--------------------------------	10
24	--------------------------------	12	24	--------------------------------	14
25	--------------------------------	00	25	--------------------------------	15
26	--------------------------------	09	26	--------------------------------	08
27	--------------------------------	05	27	--------------------------------	12
28	--------------------------------	02	28	--------------------------------	04
29	--------------------------------	14	29	--------------------------------	09
30	--------------------------------	11	30	--------------------------------	01
31	--------------------------------	08	31	--------------------------------	02
32	--------------------------------	03	32	--------------------------------	15
33	--------------------------------	10	33	--------------------------------	05
34	--------------------------------	14	34	--------------------------------	01
35	--------------------------------	04	35	--------------------------------	03
36	--------------------------------		36	--------------------------------	

Secondly, the attacker uses the remaining free message words to merge both branches to ensure that the chaining variables in both branches are the same by computing backward from the middle. At last, the rest of the differential path in both branches are verified probabilistically by computing forward from the middle.

6.2 Finding a Starting Point

Different from the choice of the message words for merging in [11], we set m_3 free at the phase of finding a starting point and use it at the phase of merging.

Table 17. The starting point, where $m_7' = m_7 \boxplus 2^4 \boxplus 2^{15} \boxplus 2^{30}$,and $\Delta m_i = 0.(i \neq 7, 0 \leqslant i \leqslant 15)$. Note that the word messages marked in green are all fixed. Those marked in black are all free while the one marked in red is to be inserted difference in.

	X_i	$\pi_1(i)$		Y_i	$\pi_2(i)$
-4	-------- -------- -------- --------		-4	-------- -------- -------- --------	
-3	-------- -------- -------- --------		-3	-------- -------- -------- --------	
-2	-------- -------- -------- --------		-2	-------- -------- -------- --------	
-1	-------- -------- -------- --------		-1	-------- -------- -------- --------	
00	-------- -------- -------- --------	00	00	-------- -------- -------- --------	05
01	-------- -------- -------- --------	01	01	--1----- -------- -1------ ----1---	14
02	-------- -------- -------- --------	02	02	10000010 11111100 01000110 00110000	07
03	-------- -------- -------- --------	03	03	0111101n 11000000 01n10010 n0100001	00
04	-------- -------- -------- --------	04	04	1001110n 00110n01 01100011 1001n000	09
05	-------- -------- -------- --------	05	05	n1010110 0111000n 00110000 01111101	02
06	11011100 10101101 01110010 01011001	06	06	n0010001 11110un1 00uuuuuu uu1111u1	11
07	01001101 01110100 11010011 11101011	07	07	uuu00un1 n1u011nn 00000110 11011n00	04
08	001100n0 uuuuuuuu uuu1nuuu 1uuuuuuu	08	08	11u0uu11 10u1u0n1 nn0nu110 0010000u	13
09	00uun0nn 1n110nnu nnuu1011 001n10nn	09	09	01uu0111 0011u000 11000n0n n0100111	06
10	10110n11 unuun0u0 u00100nn 1100u011	10	10	011u1nnn nnn01100 10000000 0100u110	15
11	10n101nu uu11nu10 un1n0100 01011100	11	11	011u1n1n 1011011u n00n1n11 100001n1	08
12	00011u00 11n1nnnn nnnnnnnn nnn01101	12	12	10011010 11n00110 u1011000 0001111n	01
13	11111000 01111111 01000011 00010100	13	13	1u100n0u 11110100 000111u0 01010111	10
14	10010011 00110110 11101010 00010010	14	14	01001000 1n000110 0000n111 11n10000	03
15	-------- -------- -------- --------	15	15	-----u-- -1-u0--- ----1--- --u--1--	12
16	-------- -------- -------- --------	07	16	-------- -------- -------- ---0----	06
17	-------- -------- -------- --------	04	17	-------- -------- -------- ---1----	11
18	-------- -------- -------- --------	13	18	-------- -------- -------- --------	03
19	-------- -------- -------- --------	01	19	-------- -------- -------- --------	07
20	-------- -------- -------- --------	10	20	-------- -------- -------- --------	00
21	-------- -------- -------- --------	06	21	-------- -------- -------- --------	13
22	-------- -------- -------- --------	15	22	-------- -------- -------- --------	05
23	-------- -------- -------- --------	03	23	-------- -------- -------- --------	10
24	-------- -------- -------- --------	12	24	-------- -------- -------- --------	14
25	-------- -------- -------- --------	00	25	-------- -------- -------- --------	15
26	-------- -------- -------- --------	09	26	-------- -------- -------- --------	08
27	-------- -------- -------- --------	05	27	-------- -------- -------- --------	12
28	-------- -------- -------- --------	02	28	-------- -------- -------- --------	04
29	-------- -------- -------- --------	14	29	-------- -------- -------- --------	09
30	-------- -------- -------- --------	11	30	-------- -------- -------- --------	01
31	-------- -------- -------- --------	08	31	-------- -------- -------- --------	02
32	-------- -------- -------- --------	03	32	-------- -------- -------- --------	15
33	-------- -------- -------- --------	10	33	-------- -------- -------- --------	05
34	-------- -------- -------- --------	14	34	-------- -------- -------- --------	01
35	-------- -------- -------- --------	04	35	-------- -------- -------- --------	03
36	-------- -------- -------- --------		36	-------- -------- -------- --------	

Message Words	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Value	*	0x67dbd0a9	*	*	0x5cd30b65	*	0x651c397d	*
Message Words	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
Value	0x050ff865	*	0xa9f94c09	0x509bf856	0x0588c327	0x86671566	*	0xc3349b51

In this way, we can improve the successful probability of merging. However, the right branch is not fully satisfied any more, thus resulting in an uncontrolled probability in the right branch.

According to the characteristics of Q_{15} displayed in Table 4, we observe that $[Q_{15}]_{31} = 0$. According to Table 5, we can find that if $Y_{11,0} = 1$, $Y_{11,29} = 1$, $Y_{11,30} = 1$ are satisfied at the phase of finding a starting point, $Y_{15,11} = 1$ will hold with a much higher probability, thus improving the uncontrolled probability in the right branch.

By adding three more bit conditions on $Y_{11,0}$, $Y_{11,29}$, $Y_{11,30}$ and setting m_3 free, using the technique for finding a starting point in [11], we obtain a new starting point displayed in Table 17.

6.3 Probability Neglected While Computing Backward

Based on the differential path in Table 16, we know that $\Delta X_5 = 0$, $\Delta X_4 = 0$, $\Delta X_3 = 0$, $\Delta Y_1 = 0$, $\Delta Y_0 = 0$, $\Delta Y_{-1} = 0$, $\Delta Y_{-2} = 0$ while $\Delta X_8 \neq 0$, $\Delta X_9 \neq 0$, $\Delta Y_3 \neq 0$, $\Delta Y_4 \neq 0$, $\Delta Y_5 \neq 0$. At the phase of finding a starting point, $\Delta X_5 = 0$ and $\Delta Y_1 = 0$ have been satisfied. However, for the original algorithm [11] to merge both branches, the conditions that $\Delta X_4 = 0$, $\Delta X_3 = 0$, $\Delta Y_0 = 0$, $\Delta Y_{-1} = 0$, $\Delta Y_{-2} = 0$ have been neglected. We define the probability of these conditions as *neglected probability*.

According to the conditions $\Delta X_4 = 0$, $\Delta X_3 = 0$, $\Delta Y_0 = 0$, $\Delta Y_{-1} = 0$ and $\Delta Y_{-2} = 0$, we can get the following equations:

$$\begin{aligned}
0 &= (X_9^{'} \boxminus X_5^{\lll 10})^{\ggg 11} \boxminus (X_9 \boxminus X_5^{\lll 10})^{\ggg 11} \boxminus (XOR(X_8', X_7', X_6'^{\lll 10}) \\
&\quad \boxminus XOR(X_8, X_7, X_6^{\lll 10})), \\
0 &= (X_8^{'} \boxminus X_4^{\lll 10})^{\ggg 9} \boxminus (X_8 \boxminus X_4^{\lll 10})^{\ggg 9} \boxminus (m_7' \boxminus m_7), \\
0 &= (Y_5^{'} \boxminus Y_1^{\lll 10})^{\ggg 13} \boxminus (Y_5 \boxminus Y_1^{\lll 10})^{\ggg 13} \boxminus (ONX(Y_4^{'}, Y_3^{'}, Y_2^{\lll 10}) \\
&\quad \boxminus ONX(Y_4, Y_3, Y_2^{\lll 10})), \\
0 &= (Y_4^{'} \boxminus Y_0^{\lll 10})^{\ggg 11} \boxminus (Y_4 \boxminus Y_0^{\lll 10})^{\ggg 11} \boxminus (ONX(Y_3^{'}, Y_2, Y_1^{\lll 10}) \\
&\quad \boxminus ONX(Y_3, Y_2, Y_1^{\lll 10})), \\
0 &= (Y_3^{'} \boxminus Y_{-1}^{\lll 10})^{\ggg 9} \boxminus (Y_3 \boxminus Y_{-1}^{\lll 10})^{\ggg 9} \boxminus (m_7' \boxminus m_7).
\end{aligned}$$

Observing the five equations above, it is easy to find that there are some similarities between them. Therefore, we can change the problem of calculating the probability that the five equations hold into calculating the probability that T satisfies $(T \boxplus C_0)^{\ggg S} = T^{\ggg S} \boxplus C_1$. Let $T' = T^{\ggg S}$, the equation becomes $T'^{\lll S} \boxplus C_0 = (T' \boxplus C_1)^{\lll S}$, whose probability can be calculated as introduced before.

For equation $(X_9^{'} \boxminus X_5^{\lll 10})^{\ggg 11} \boxminus (X_9 \boxminus X_5^{\lll 10})^{\ggg 11} \boxminus (XOR(X_8', X_7', X_6'^{\lll 10}) \boxminus XOR(X_8, X_7, X_6^{\lll 10})) = 0$, $X_9' \boxminus X_9 = \texttt{0xdb459013}$, $XOR(X_8', X_7', X_6'^{\lll 10}) \boxminus XOR(X_8, X_7, X_6^{\lll 10}) = \texttt{0x25b68b3}$, $C_0 = \texttt{0xdb459013} \boxminus 0 = \texttt{0xdb459013}$, $C_1 = \texttt{0x25b68b3}$. Therefore, $P(\Delta X_4 = 0) = P(T^{\lll 11} \boxplus \texttt{0xdb459013} = (T \boxplus \texttt{0x25b68b3})^{\lll 11}) \approx 2^{-11.7}$. In the same way, we can obtain that

$$\begin{aligned}
P(\Delta X_3 &= 0) = P(T^{\lll 9} \boxplus \texttt{0x1002081} = (T \boxplus \texttt{0x40008010})^{\lll 9}) \approx 2^{-8.4},\\
P(\Delta Y_0 &= 0) = P(T^{\lll 13} \boxplus \texttt{0x80010000} = (T \boxplus \texttt{0xfffc0008})^{\lll 13}) =\approx 2^{-1},\\
P(\Delta Y_{-1} &= 0) = P(T^{\lll 11} \boxplus \texttt{0x1040008} = (T \boxplus \texttt{0x1002080})^{\lll 11}) \approx 1,\\
P(\Delta Y_{-2} &= 0) = P(T^{\lll 9} \boxplus \texttt{0x1002080} = (T \boxplus \texttt{0x40008010})^{\lll 9}) \approx 2^{-0.4}.
\end{aligned}$$

Therefore, the *negelected probability* is $2^{-11.7-8.4-1-0.4} = 2^{-21.5}$. In order to eliminate the influence of the *negelected probability* at the phase of merging, for a given starting point, we can pre-compute the valid m_9 that makes $\Delta X_4 = 0$ and $\Delta X_3 = 0$ satisfied, which costs 2^{32} time and about $2^{32} \times P(\Delta X_4 = 0) \times P(\Delta X_3 = 0) = 2^{32-11.7-8.4} = 2^{12.9}$ memory. Then, at the phase of merging, given one valid m_9, we can firstly compute and store the valid m_2 that makes $Y_{1,3} = 1$, $Y_{1,14} = 1$, $Y_{1,29} = 1$, $\Delta Y_0 = 0$ and $\Delta Y_{-1} = 0$ satisfied, which costs 2^{29} time and about $2^{29} \times P(\Delta Y_0 = 0) \times P(\Delta Y_1 = 0) = 2^{28}$ memory. After choosing the valid m_9 and m_2, only the condition $\Delta Y_{-2} = 0$ has an influence on the merging, whose probability is $P(\Delta Y_{-2} = 0) \approx 2^{-0.4}$.

6.4 Merging both Branches with $m_0, m_2, m_3, m_5, m_7, m_9, m_{14}$

At the merging phase, our target is to use the remaining free message words to obtain a perfect match on the values of the five initial chaining variables of both branches. Our procedure of merging is detailed as below.

Step 1: Choose a valid value of m_9, then compute until X_4 in the left branch. Fix $Y_{1,3} = 1$, $Y_{1,14} = 1$, $Y_{1,29} = 1$ and exhaust all the 2^{29} possible values of Y_1. Then compute and store the valid m_2 that makes $\Delta Y_0 = 0$, $\Delta Y_1 = 0$ satisfied. We denote the valid number of m_2 by $VNUM$ and define the array that stores the valid m_2 as $VALIDM2[]$.

Step 2: Set random values to m_7, then compute until X_2 in the left branch.

Step 3: Set $m_2 = VALIDM2[index]$ (initialize *index* as 0), Y_1 and Y_0 can be computed based on the following equation. If *index* becomes $VNUM$ again, goto Step 2.

$$\begin{aligned}
Y_1^{\lll 10} &= (Y_6 \boxminus Y_2^{\lll 10})^{\ggg 15} \boxminus (ONX(Y_5, Y_4, Y_3^{\lll 10}) \boxplus m_2 \boxplus K_0^r),\\
Y_0^{\lll 10} &= (Y_5 \boxminus Y_1^{\lll 10})^{\ggg 13} \boxminus (XOR(Y_4, Y_3, Y_2^{\lll 10}) \boxplus m_9 \boxplus K_0^r).
\end{aligned}$$

Step 4: Since $X_0 = Y_0$ and we have obtained the value of Y_0 at Step 3, we can compute X_0, X_1 and m_5 as follows. $X_0 = Y_0$, $X_1^{\lll 10} = X_5 \boxminus (X_0^{\lll 10} \boxplus ONX(X_4, X_3, X_2^{\lll 10}) \boxplus m_4 \boxplus K_0^l)^{\lll 5}$, $m_5 = (X_6 \boxminus X_2^{\lll 10})^{\ggg 8} \boxminus (X_1^{\lll 10} \boxplus ONX(X_5, X_4, X_3^{\lll 10}) \boxplus K_0^l)$.

Step 5: We can use the conditions $X_{-1} = Y_{-1}$ and $X_{-2} = Y_{-2}$ to construct an equation system of m_0 and m_3. Observe the step functions:

$$\begin{aligned}
X_{-1}^{\lll 10} &= (X_4 \boxminus X_0^{\lll 10})^{\ggg 12} \boxminus (XOR(X_3, X_2, X_1^{\lll 10}) \boxplus m_3 \boxplus K_0^l),\\
X_{-2}^{\lll 10} &= (X_3 \boxminus X_{-1}^{\lll 10})^{\ggg 15} \boxminus (XOR(X_2, X_1, X_0^{\lll 10}) \boxplus m_2 \boxplus K_0^l),\\
Y_{-1}^{\lll 10} &= (Y_4 \boxminus Y_0^{\lll 10})^{\ggg 11} \boxminus (ONX(Y_3, Y_2, Y_1^{\lll 10}) \boxplus m_0 \boxplus K_0^r),\\
Y_{-2}^{\lll 10} &= (Y_3 \boxminus Y_{-1}^{\lll 10})^{\ggg 9} \boxminus (ONX(Y_2, Y_1, Y_0^{\lll 10}) \boxplus m_7 \boxplus K_0^r).
\end{aligned}$$

Let $A = (X_4 \boxminus X_0^{\lll 10})^{\ggg 12} \boxminus (XOR(X_3, X_2, X_1^{\lll 10}) \boxplus K_0^l)$, $B = (Y_4 \boxminus Y_0^{\lll 10})^{\ggg 11} \boxminus (ONX(Y_3, Y_2, Y_1^{\lll 10}) \boxplus K_0^r)$, $C = XOR(X_2, X_1, X_0^{\lll 10}) \boxplus m_2 \boxplus K_0^l$, $D = ONX(Y_2, Y_1, Y_0^{\lll 10}) \boxplus m_7 \boxplus K_0^r$, $T' = X_3 \boxminus A \boxplus m_3$, $T = T'^{\ggg 15}$, $C_0 = Y_3 \boxminus X_3$, $C_1 = D \boxminus C$. According to the condition $X_{-1} = Y_{-1}$, we can obtain one equation: $A \boxminus m_3 = B \boxminus m_0$. According to the condition $X_{-2} = Y_{-2}$, we can obtain another equation: $T^{\lll 15} \boxplus C_0 = (T \boxplus C_1)^{\lll 9}$. As introduced before, we can obtain its solutions by 2^2 computations on average. If there is no solution, goto Step 3. It is essential that all solutions should be taken into consideration since there may be more than one solution to the equation $T^{\lll 15} \boxplus C_0 = (T \boxplus C_1)^{\lll 9}$.

Step 6: Compute X_{-1} and Y_{-1} by m_3. Since $\Delta Y_{-2} = 0$ holds with probability, we have to check whether Y_{-1} satisfies the equation $0 = (Y_3' \boxminus Y_{-1}^{\lll 10})^{\ggg 9} \boxminus (Y_3 \boxminus Y_{-1}^{\lll 10})^{\ggg 9} \boxminus (m_7' \boxminus m_7)$. If this equation doesn't hold for all pairs of $(m_0,\, m_3)$, goto Step 3.

Step 7: Compute $X_{-2},\ Y_{-2},\ X_{-3},\ Y_{-3}$ and m_{14}.

Step 8: This is the uncontrolled part of merging. At this point, all freedom degree have been used and the last condition $X_{-4} = Y_{-4}$ will hold with probability 2^{-32}.

Verification. We have verified the merging phase by implementation. Based on the starting point in Table 17, we choose a valid value of $m_9 = \texttt{0x471fba32}$, and the number of the corresponding valid m_2 is `0xfcf2100`. The following is an instance obtained by carrying out the merging phase.

$$m_0 = \texttt{0x678c8c36}, m_2 = \texttt{0x5293b823}, m_3 = \texttt{0xd90c1aa9}, m_5 = \texttt{0x13d3dff6},$$
$$m_7 = \texttt{0x794a60c6}, m_{14} = \texttt{0xee8e443e}, Y_{-4} = \texttt{0xd055ce6}, Y_{-3} = \texttt{0xdf979ac7},$$
$$Y_{-2} = \texttt{0xae4836b3}, Y_{-1} = \texttt{0x57b6f5fb}, Y_0 = \texttt{0x6b9ec934}.$$

6.5 Uncontrolled Probability

Firstly, we give the theoretical calculation of the uncontrolled probability of the left branch.

$$\begin{aligned} P(\Delta X_{15} = 0) &= P(T^{\lll 9} \boxplus \texttt{0xf0bfff7f} = (T \boxplus \texttt{0x7f785fff})^{\lll 9}) \\ &= (2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-9} + \Sigma_{i=11}^{23} 2^{-i}) \times (1 - 2^{-1} - \Sigma_{i=3}^{9} 2^{-i}) \\ &= \frac{\texttt{0x3ca85f7f}}{2^{32}} \approx 2^{-2.1}, \\ P(\Delta X_{16} = 0) &= P(T^{\lll 8} \boxplus \texttt{0x40008010} = (T \boxplus \texttt{0xf400081})^{\lll 8}) \\ &= 2^{-4} \times (2^{-2} + 2^{-17} + 2^{-24}) \\ &= \frac{\texttt{0x4000810}}{2^{32}} \approx 2^{-6}. \end{aligned}$$

Therefore, the theoretical value of the uncontrolled probability of the left branch is about $2^{-2.1-6} = 2^{-8.1}$.

Secondly, we use our method to evaluate the uncontrolled probability of the right branch. Since we add three bit conditions on Y_{11} at the phase of finding a starting point, it is necessary to fix the values of the three bits before calculation. Then, we obtain that the probability that the modular difference of Y_{15} and the seven bit conditions (as showed in Table 16) on Y_{15} hold is about $2^{-7.6}$. The probability that the modular difference of Y_{16} and the one bit condition (as showed in Table 16) on Y_{16} hold is about 2^{-2}. The probability that the modular difference of Y_{17} and the one bit condition (as showed in Table 16) on Y_{17} hold is about 2^{-2}. The probability that the modular difference of Y_{18} holds is about 1. The probability that the modular difference of Y_{19} holds is about $2^{-0.4}$. Besides, there are five more bit conditions on Y_{15}, Y_{16} and Y_{17}, which are $Y_{15,0} = Y_{16,0}$, $Y_{15,15} = Y_{16,15}$, $Y_{15,21} = Y_{16,21}$, $Y_{16,15} = Y_{17,15}$ and $Y_{16,30} = Y_{17,30}$. Therefore, with our method to calculate the step differential probability, the uncontrolled probability of the right branch is about $2^{-7.6-2-2-0.4-5} \approx 2^{-17}$.

Then, we consider the uncontrolled probability of both branches for a specific starting point in Table 17. We can calculate the uncontrolled probability of the left branch in this way: exhaust all 2^{32} possible values of m_{14} and count the number of m_{14} which makes $\Delta X_{15} = 0$ and $\Delta X_{16} = 0$ satisfied. According to the experiment, the valid number of m_{14} is `0x1020000` and thus the uncontrolled probability of the left branch is about 2^{-8}. For the uncontrolled probability of the right branch, we can exhaust all 2^{32} possible values of m_3 and count the number of m_3 which makes the conditions on Y_{15}, Y_{16}, Y_{17}, Y_{18}, Y_{19} satisfied. According to the experiment, the valid number of m_3 is `0x9f64` and thus the uncontrolled probability of the right branch is about $2^{-16.68}$. We have to stress this is the uncontrolled probability of both branches for a specific starting point. Comparing this result with the theoretical value, we observe that they are almost the same, which implies that our method to theoretically calculate the step differential probability is reliable.

Moreover, during the merging phase, we can not control the value matching on the first IV word, and it adds another factor 2^{-32}. Since the expected value of the number of the solution to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ is 1, its influence on the probability can be ignored. What's more, $Y'_{-2} = Y_{-2}$ holds with probability $2^{-0.4}$. Therefore, the total uncontrolled probability is $2^{-32-8.1-17-0.4} = 2^{-57.5}$, which is much higher than the original one $2^{-72.6}$. Given a starting point, the degree of freedom left is 32+28+12=72 since m_7, m_2, m_9 can take 2^{32}, 2^{28}, 2^{12} possible values respectively. Besides, we can generate many staring points to mount the semi-free-start collision attack on the first 36-step RIPEMD-160. Therefore, the degree of freedom is enough.

6.6 Complexity Evaluation

Firstly, we consider the complexity of the merging phase. Based on the fact that $X_{-4} = Y_{-4}$ holds with probability 2^{-32}, $Y'_{-2} = Y_{-2}$ holds with probability $2^{-0.4}$, and the expectation of the number of the solution to $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$ is 1, we can give an estimation of the running times of each step at the merging phase. We estimate that Step 7 to Step 8 will run for 2^{32} times, Step 6 will run for $2^{32+0.4} = 2^{32.4}$ times, Step 3 to Step 5 will run for $2^{32+0.4} = 2^{32.4}$ times, Step 2 will run for $2^{32.4-28} = 2^{4.4}$ times, Step 1 will run for only one time. Since Step 2 contains about 2-step computation of the step function, Step 3 to

Step 5 contains about ($8{+}2^2{=}12$)-step computation of the step function, Step 6 contains about 2-step computation of the step function, and Step 7-8 contains 5-step computations of the step function, we estimate the complexity of the merging phase as $2^{4.4} \times 2/72 + 2^{32.4} \times 12/72 + 2^{32.4} \times 2/72 + 2^{32} \times 5/72 \approx 2^{30}$. Taking the uncontrolled probability of both branches into consideration, the complexity becomes $2^{30+17+8.1} = 2^{55.1}$.

Next, we consider the memory complexity of the merging phase. Given a valid m_9, computing the valid values of m_2 and storing the results costs 2^{29} time and 2^{28} memory. At the pre-computing phase, pre-computing the valid values of m_9 and storing the results costs 2^{32} time and $2^{12.9}$ memory. In addition, pre-computing the equation $T^{\lll 15} \boxplus C_0 = T^{\lll 9}$ costs 2^{32} time and 2^{32} memory. Since the probability of the 36-semi-free-start collision attack is $2^{-57.5}$, one valid m_9 is enough for the improved attack. Therefore, at the merging phase, the memory complexity is $2^{32} + 2^{28}$. Since the time complexity of computing valid m_2, m_9 and pre-computing the equation is much smaller than $2^{55.1}$, it can be ignored. In summary, the time complexity of the semi-free-start collision attack on RIPEMD-160 reduced to 36 steps is $2^{55.1}$ and the memory requirements are $2^{32} + 2^{28} + 2^{12.9} \approx 2^{32}$. The implementation of this attack is available at https://github.com/Crypt-CNS/RIPEMD160-36Step.git.

7 Conclusion

In this paper, we propose a feasible method to theoretically calculate the step differential probability, which was stated as an open problem at ASIACRYPT 2013. Besides, we propose a method to reduce the time complexity and memory complexity to pre-compute the equation $T^{\lll S_0} \boxplus C_0 = (T \boxplus C_1)^{\lll S_1}$. Based on our analysis of the expectation of the number of the solutions to this equation, we conclude that our new way to obtain the solutions only costs four times of checking. In addition, we construct a differential path where the left branch is sparse and the right branch is controlled as sparse as possible. Using the message modification techniques and deducing some extra bit conditions based on the equation that Q_i has to satisfy, it is possible to mount a 30-step collision attack on RIPEMD-160 with probability about 2^{-67}. What's more, based on the 36-step differential path found by Mendel *et al.*, we take a different strategy to choose the message words for merging. In this way, we improve the time complexity of the semi-free-start attack on the first 36-step RIPEMD-160. Compared with the best analytical result of this attack on RIPEMD-160, we reduce the time complexity from $2^{70.4}$ to $2^{55.1}$. Moreover, our improvement also brings us some insights into the choice of message words for merging. Therefore, the message words for merging should be determined with care, which will make a difference.

Acknowledgements. The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. Fukang Liu and Gaoli Wang are supported by the National Natural Science Foundation of China (Nos. 61572125, 61632012, 61373142), and Shanghai High-Tech Field Project (No. 16511101400). Florian Mendel has been supported by the Austrian Science Fund (FWF) under grant P26494-N15.

References

1. Bosselaers, A., Preneel, B. (eds.): RIPE 1992. LNCS, vol. 1007. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60640-8
2. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
3. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: a strengthened version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60865-6_44
4. Dobbertin, H.: RIPEMD with two-round compress function is not collision-free. J. Cryptol. **10**(1), 51–69 (1997)
5. Landelle, F., Peyrin, T.: Cryptanalysis of full RIPEMD-128. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 228–244. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_14
6. Daum, M.: Cryptanalysis of hash functions of the MD4-family (2005). http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/DaumMagnus/diss.pdf
7. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_16
8. Mendel, F., Nad, T., Schläffer, M.: Collision attacks on the reduced dual-stream hash function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_14
9. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential attacks on reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 23–38. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33383-5_2
10. Mendel, F., Nad, T., Schläffer, M.: Improving local collisions: new attacks on reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 262–278. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_16
11. Mendel, F., Peyrin, T., Schläffer, M., Wang, L., Wu, S.: Improved cryptanalysis of reduced RIPEMD-160. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 484–503. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_25
12. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_40
13. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage attacks on step-reduced RIPEMD-128 and RIPEMD-160. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 169–186. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21518-6_13
14. Sasaki, Y., Wang, L.: Distinguishers beyond three rounds of the RIPEMD-128/-160 compression functions. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 275–292. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_17

15. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 570–596. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_19
16. Wang, G., Wang, M.: Cryptanalysis of reduced RIPEMD-128. J. Softw. **19**(9), 2442–2448 (2008)
17. Wang, G.: Practical collision attack on 40-step RIPEMD-128. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 444–460. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_23
18. Wang, G., Yu, H.: Improved cryptanalysis on RIPEMD-128. IET Inf. Secur. **9**(6), 354–364 (2015)
19. Wang, G., Shen, Y., Liu, F.: Cryptanalysis of 48-step RIPEMD-160. IACR Trans. Symmetric Cryptol. **2017**(2), 177–202 (2017)
20. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_1
21. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_2
22. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_1
23. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_2